

Efficient Clustering of Metagenomic Sequences using Locality Sensitive Hashing

Zeehasham Rasheed * Huzefa Rangwala † Daniel Barbará‡

Abstract

The new generation of genomic technologies have allowed researchers to determine the collective DNA of organisms (e.g., microbes) co-existing as communities across the ecosystem (e.g., within the human host). There is a need for the computational approaches to analyze and annotate the large volumes of available sequence data from such microbial communities (metagenomes).

In this paper, we developed an efficient and accurate metagenome clustering approach that uses the locality sensitive hashing (LSH) technique to approximate the computational complexity associated with comparing sequences. We introduce the use of fixed-length, gapless subsequences for improving the sensitivity of the LSH-based similarity function. We evaluate the performance of our algorithm on two metagenome datasets associated with microbes existing across different human skin locations. Our empirical results show the strength of the developed approach in comparison to three state-of-the-art sequence clustering algorithms with regards to computational efficiency and clustering quality. We also demonstrate practical significance for the developed clustering algorithm, to compare bacterial diversity and structure across different skin locations.

Keywords: sequence clustering, metagenomics, locality sensitive hashing

1 Introduction

Advances in sequencing technologies have allowed researchers to determine the collective genomes of microbes co-existing as communities across different ecological and human-host environments [1]. This process of sequencing the microbial communities in a collective fashion is referred to as “metagenomics” (see Background Section 2 for details). Determining the content, abundance and functionality of the different microbes within the human-host samples is critical for understanding the pathogenic or symbiotic role played by these microbes.

Sequencing technologies produce short, contiguous fragments (called reads) from random positions of the long genomes. These reads need to be stitched (assembled) together based on the overlap between different reads. In the case of metagenomic sequencing, the challenge of determining the genomes for multiple bacterial species increases several folds. Several computational

methods have been developed to tackle the challenges associated with metagenomic analysis [2].

Clustering methods have been developed for handling the output of metagenomic sequencing projects, which contain a large set of sequence reads of unknown origin. Clustering approaches like CD-HIT [3], UCLUST [4] and CROP [5] put similar sequences in the same groups (maybe species-specific) and then cluster representatives can be used for the rapid analysis of species diversity within different metagenome samples. Different tools such as DOTUR [6], Mothur [7] and ESPRIT [8] improve the process of comparing metagenome samples by using hierarchical clustering to partition the input data into clusters.

In this work, we propose a new, scalable metagenomic sequence clustering algorithm called MC-LSH. The key characteristic of our algorithm is the use of an efficient randomized search technique called “locality sensitivity hashing” (LSH) [9]. LSH has also been used in similarity searching [10], malware clustering [11] and hierarchical clustering [12]. Similarity among the sequences is computed based on randomly chosen indices that essentially compresses our input sequences. Buhler et. al. [13] used the LSH-based approach to determine all pairs of similar segments between two long genomes. We incorporate the use of fixed-length gapless subsequences, commonly referred to as w -mer to improve the sensitivity of matching pairs of sequences. Using w -mers per index position helps in the identification of conserved regions, and improves the accuracy of comparing sequence pairs. The clustering algorithm follows a greedy, iterative approach that is optimized to work for equal and unequal length sequence sets.

We perform a comprehensive set of experiments evaluating the performance of MC-LSH on two different metagenomic datasets associated with the microbes existing across different human skin locations [14, 15] Our benchmarking focuses on the quality of clustering evaluated using external ground truth, computational complexity (run time and memory usage) and pairwise sequence similarity of sequences within a cluster. Our results demonstrate the strength of MC-LSH in comparison to state-of-the-art sequence clustering algorithms.

We also demonstrate a use-case scenario for MC-LSH. The algorithm is used to compare the diversity

*Department of Computer Science, George Mason University.
Email: zrasheed@gmu.edu

†Department of Computer Science, George Mason University.
Email: rangwala@cs.gmu.edu

‡Department of Computer Science, George Mason University.
Email: dbarbara@gmu.edu

of bacterial species across different skin locations. The hypothesis supported by our down stream analysis is also supported by independent scientific findings.

2 Background and Motivation

2.1 Metagenomics Advances in genomic technologies have equipped researchers with the capacity to determine the collective DNA/genome of entire co-existing microbial communities, omnipresent across the ecosystem i.e., sea [16], soil and human body [17]. In fact, the human body contains one of the most densely populated microbial environments known on earth where over 10^{14} microbial cells interact with ours, indicating the ubiquity and relevance of these interactions with regards to health and disease [18]. This collective genomic experiment is defined as “metagenomics”, and provides an unbiased view of the diversity and biological potential of these communities [2].

In the same context, 16S rRNA gene sequencing has been widely used for the analysis of genetic diversity of complex bacterial communities [19]. 16S sequences are marker genes i.e., they are part of most microbial species but have variations in their sequence that allows them to be separated into different bacterial groups [20]. Metagenomic projects follow sequencing of 16S genes (rather than entire genomes) as the first step in identifying and profiling the different communities.

Metagenomic projects pose several computational challenges related to the assembly (determination of multiple, complete genomes from the input of sequencing machines), identification of the different species, annotation and correlation to metadata [2]. Specifically, microbial communities (especially within the human host) are complex, have varying genome lengths, varying abundances, and have never been seen before in reference genome databases. Further, the sequencing technologies have their idiosyncrasies with regards to errors, sequence read lengths and volume of sequences produced (runs into millions).

Several computational approaches have been developed to compare pairs of metagenome samples [7, 6], cluster metagenome sequences within a metagenome sample [4, 3, 5] and classify metagenome samples into different phylogenetic or taxonomic groups [21]. Clustering of metagenomic samples reduce the complexity associated with analyzing every DNA sequence within a sample. Clustering can lead to groupings that are species-specific and hence, assists in the identifying the content and abundance of microbial species within the metagenome samples. CD-HIT [3], UCLUST [4] and CROP [5] are state-of-the-art sequence clustering approaches that are used extensively for clustering metagenomic samples, often as the first step for bioinformatic

analysis. Below, we provide a brief description of these three approaches:

2.2 UCLUST. UCLUST [4] is a greedy clustering algorithm that achieves its computational efficiency by using gapless, high-scoring common subsequences or segments called as HSPs to compare pairs of sequences. At first, the algorithm identifies exact, gapless, common matches of fixed length (seeds) between sequence pairs and then extends them by allowing for spaces and mismatches between the alignments. Only the seeds having score above a fixed threshold are chosen as valid HSPs.

The algorithm follows an incremental approach to assigns sequences in the input to different clusters. Initially, the clustering solution is maintained as an empty list. Sequences are incrementally added to existing clusters, or used to create new clusters. Each input sequence is compared to the existing cluster representatives using a fast search procedure (USEARCH), that uses the HSPs. If the input sequence finds a matching sequence with one of the cluster representatives, then the sequence is assigned to that particular cluster, otherwise the input sequence forms a new cluster (with itself being the representative). As a pre-processing step, UCLUST requires the input sequences to be sorted in decreasing order of their length. This ensures that the cluster representative is always the largest sequence.

2.3 CD-HIT. CD-HIT [3] is identical in operation to the UCLUST algorithm. Sequences are added to clusters in an incremental, greedy manner. Comparisons are always performed to cluster representatives. CD-HIT differs in UCLUST in the method used to compare pairs of sequences. Short gapped subsequences are used to determine if pairs of sequences are similar according to a preset threshold. If the heuristic, cannot confirm the similarity, an exact sequence alignment is performed.

2.4 CROP. CROP [5] follows a different approach in comparison to UCLUST and CD-HIT. It uses a Gaussian mixture model to describe the data and is developed for 16S rRNA marker sequences only. A “center” sequence is used to characterize a specific cluster and the probability that a sequence belongs to a cluster is defined as a function of the distance between the sequence and representative center. Based on Gaussian distributions model, the likelihood of sequence data can be defined. A MCMC-based approach is used to sample from the posterior distribution of the parameters to obtain the optimal clustering solution. The optimal result maximizes the posterior probability, optimizing

the number of clusters, their relative abundance levels and the sequences in each cluster. To enhance computational efficiency, CROP uses a hierarchical approach by splitting the data into small blocks and running Bayesian clustering on each block independently, and then later merging these clustering results.

3 Methods

In this section, we present our approach to cluster the metagenome sequences using a locality sensitive hashing (LSH) based function. The LSH function allows us to reduce the complexity of exact pairwise string matching or sequence alignment. We refer to our approach as MC-LSH (Metagenomic Clustering using LSH). LSH was developed by Indyk et. al. [9], and later Gionis et. al. [22] modified it for solving high-dimensional computational geometry problems such as finding approximate nearest neighbors. Buhler [13] used the LSH approach for determining all pairs of local alignments, given a pair of long genomes. We first review the principles of LSH and then provide a detailed description of the MC-LSH algorithm.

3.1 Locality Sensitivity Hashing Given a nucleotide string s of length n , we construct a randomized hash function. We choose k uniform, random indices i_1, \dots, i_k in the range $\{1, \dots, n\}$ to define a hash function $f(s)$ given by:

$$(3.1) \quad f(s) = \langle s[i_1], s[i_2], \dots, s[i_k] \rangle$$

The function $f(s)$, given in Equation 3.1 produces a concatenated string of length k (the number of chosen random indices) from the original string s of length n . DNA sequences are made up of four nucleotides, and the function $f(s)$ creates a mapping between the original 4^n dimensional space to a reduced 4^k dimensional space. As described by Indyk et. al. [9], function $f(s)$ is called “locality-sensitive” because the probability that a pair of input strings s_1 and s_2 have the same hash value varies directly with their similarity. If the strings s_1 and s_2 are similar i.e., assumed to be differing by at the most r nucleotides, then the probability that they produce the same hash values is given by:

$$(3.2) \quad P[f(s_1) = f(s_2)] \geq \left(1 - \frac{r}{n}\right)^k.$$

Parameter r is the allowable mismatch factor between the two strings and the probability, $P[\]$ is computed over all random choices of the k indices, i_1, \dots, i_k . The LSH function also captures the partial global ordering of nucleotides that are present in the DNA sequences.

We use the hamming distance function denoted by $H(\cdot, \cdot)$, to compute the difference between the hash values, $f(s_1)$ and $f(s_2)$ for comparing the pairs of strings s_1 and s_2 , respectively. Based on the distance function, we can accept pairs of strings that differ by only a few substitutions, while reject pairs that differ by many substitutions. String pairs that match exactly are always be accepted by this function filter.

However, the use of hash function and hamming distance can lead to both false positives and false negatives. A false positive occurs when strings, s_1 and s_2 are dissimilar but hash to the same values i.e., $f(s_1) = f(s_2)$. This occurs because the function f samples those k indices/positions, that have the same nucleotides (or characters) in the two strings. A false negative occurs when strings s_1 and s_2 are similar (not 100% identical) but have $f(s_1) \neq f(s_2)$, because f samples one or more of the k indices where s_1 and s_2 differ. False negatives cannot be detected easily, but can be reduced substantially with multiple repeats of the filtering process. This is done by repeatedly sampling new hash functions i.e., new sets of k indices. In this paper, the number of times or iterations for which the k indices will be sampled is referred by l .

Instead of using a single nucleotide (character) at the randomly chosen index i , we use a gapless subsequence of fixed length per index for our hash function. We refer to the subsequence of length w , starting at position $i \in 1 \dots n-w$ of sequence s as w -mer(i). Using the w -mer allows the filtering function to be more exact and reduces false positives, because now we compare w -mers at the k different indices. DNA and protein sequences have local sequential dependencies which can be captured using short, gapless subsequences. As such, the use of w -mers, also referred to as N -grams or k -mers is prevalent in several bioinformatic applications, including sequence assembly, error correction for sequencing [23], motif detection and sequence classification [24].

3.2 MC-LSH Parameters. The MC-LSH algorithm has the following parameters: (i) k , the number of sampled indices for the hash key, (ii) l , the number of iterations of hash functions, (iii) w , the length of subsequence and r , the mismatch factor. These parameters influence the clustering results in terms of accuracy, quality and also influence the running time and memory usage of our algorithm.

3.2.1 Number of sampled indices (k) and subsequence length w : The hash function provides a mapping from the 4^n -dimensional space into 4^k -dimensional space. Using small values for k will lead to small number of partitions, and large number of false positives,

since the number of sampled bits or indices will not be enough to disambiguate dissimilar reads.

The use of gapless, subsequence of length w enhances the filtering quality. Instead of comparing a single nucleotide at a given index position, we compare a w -mer at every chosen index. For pairs of strings to hash to the same key value, requires an exact matching of $k * w$ nucleotides. This makes the matching process more stringent and will lead to a reduction in the number of false positives.

3.2.2 Number of hash functions or iterations

(l): The use of multiple hash functions within the MC-LSH reduces the number of false negatives i.e., two sequences that are similar will have a higher chance of being accepted by our filter due to repeated sampling of k indices. The results of multiple iterations is combined using a union set operation i.e., a pair of strings are considered to have the hamming distance between them zero, as long as we see the strings hash to the same values in one of the l iterations (or l samplings). However, using large values of l will lead to dissimilar sequences being mapped to the same hash value and will lead to an increased number of false positives. A good choice of l and k will allow the MC-LSH algorithm to produce sensitive partitioning of the sequence data.

3.2.3 Percentage of mismatches (r):

In order to allow for pairs of strings that are not exactly identical, but similar (i.e., a few nucleotide mismatches) to pass the LSH-based hamming distance filter, the mismatch factor parameter r is implemented as the percentage of allowable mismatches. When r is set to 0%, the LSH-based function will consider strings to be equivalent if and only if all the k nucleotides or k w -mers are exactly identical in both the strings. As an example, when k is set to 64 indices and r is set to 10% mismatch, then pairs of strings that differ by at most 6 nucleotides or 6 w -mers will be considered to be equivalent. In our current approach, each w -mer can be considered a symbol. We do not allow for mismatches within a w -mer and is considered as a part of the future directions of this work.

3.3 Algorithm Details.

In this section we describe the design of our MC-LSH clustering algorithm. First, we describe our algorithm for handling DNA sequences of equal length and then show the extensions needed for handling unequal length sequences. Some of the genomic technologies produce sequence reads of equal lengths, whereas some produce sequence reads of varying lengths. As such, we evaluate the performance of

MC-LSH algorithm for both these scenarios.

In Algorithm 1, we provide the pseudocode for clustering a set of N sequences, \mathcal{S} , each having equal length n . For the sake of simplicity, we describe the procedure for one iteration i.e., setting l parameter to 1. The input parameters for this procedure include the number of sampled indices k , w -mer length and r , the percentage mismatch factor. We use the LSH function $f(s)$ for a given string. $f()$ uses the k indices and the w -mers at the chosen k positions to determine the hash value for a given string. The hamming distance function $H(,)$ is used to compute the distance between a pair of hashed values.

The MC-LSH algorithm is greedy in nature. After initializing the LSH-function, f by choosing k random indices (Step 2) we compute the hash values for all the sequences in the set, \mathcal{S} . We begin by choosing the first sequence (could be any one in the set) and assign the sequence to be part of the first cluster. Using the LSH-based hamming distance function, we identify all sequences in \mathcal{S} (that do not have a cluster assignment), which have hash values that differ by at most r percentage of k (Step 10). All these set of sequences belong to the same cluster and will be removed from \mathcal{S} . We iterate through Steps 5-14 for all all sequences in set \mathcal{S} that do not have a cluster assignment.

To improve the ability of our LSH-based function to identify similar sequences, we repeat the procedure described in Algorithm 1 multiple times i.e., l times. For every iteration, a new set of k indices are randomly chosen, giving us a new LSH function at each iteration. We proceed by performing a union operation of the cluster assignments obtained for the current iteration with the previous iteration. Our implementation is optimized, so as to merge clusters from the previous iteration based on the similarity obtained using the new set of k indices. For example, if sequences s_x and s_y were assigned to two different clusters in the previous iteration, and were found to be similar in the current iteration then the clusters would be merged (unionized).

3.4 Unequal length sequences.

The MC-LSH algorithm described above assumes that the sequences provided as input are of equal length. Equal length sequences limit the range of sampled k bits.

To handle sequences of unequal lengths, we determine the sequence with the shortest length n_{min} . Given, a sequence in the input set of length n we compute the LSH values for all $n - n_{min}$ subsequences each of the same length n_{min} . To compare a pair of strings, we use the hamming distance between all pairs of subsequences of length n_{min} generated from the two strings. To speed

Algorithm 1 MC-LSH algorithm for $l = 1$

Input: A set of N sequences $\mathcal{S} = \{s_1 \dots s_N\}$ with length n .

Parameters: k , w and r

Functions: $H(\cdot, \cdot)$ computes the Hamming Distance between a pair of strings.

$f(\cdot)$ computes the LSH value for an input string.

Output: Cluster Assignments, indexed by different sequences $C[s_1 \dots s_N]$

- 1: Initialize array C , $\forall s_x \in \mathcal{S}$, $C[s_x] = -1$.
 - 2: Initialize LSH function $f(\cdot)$ by choosing k random indices $i_1 \dots i_k \in 1 \dots n$.
 - 3: $\forall s \in \mathcal{S}$, compute LSH-value $f(s)$
 - 4: **repeat**
 - 5: Choose $s_x \in \mathcal{S}$, such that $C[s_x] == -1$ i.e., s_x is not assigned
 - 6: Assign $C[s_x]$ to a **new** cluster label.
 - 7: Remove s_x from \mathcal{S}
 - 8: **for** $\forall s_y \in \mathcal{S}$ **do**
 - 9: */* For all s_y in \mathcal{S} that are not assigned, try to assign these sequences the same cluster label */*
 - 10: **if** $H(f(s_x), f(s_y)) \leq \lfloor (r/100) \rfloor * k$ **then**
 - 11: $C[s_y] \leftarrow C[s_x]$
 - 12: */* s_x and s_y are in the same cluster */*
 - 13: Remove s_y from \mathcal{S}
 - 14: **end if**
 - 15: **end for**
 - 16: **until** \mathcal{S} is empty and all sequences are assigned.
 - 17: **return** C
-

up the computation, we use a greedy approach, where the two strings will be inferred to hash to the same value and belong to the same cluster, if any pair of n_{min} subsequences satisfy the filtering condition as defined by Step 10 in Algorithm 1.

4 Experimental Evaluation.

4.1 Dataset Description. We perform evaluation of our MC-LSH clustering algorithm on two different human skin microbiome datasets, previously studied by Grice et. al. [15]. The first dataset contains 1130 full-length 16S metagenomic sequences extracted from microbiome at a specific skin location called “auxiliary vault”. For the evaluation of our clustering results, we use external labels that are taxonomy class labels for the different sequences. These taxonomy labels were used in a previous study of Hao et. al. [5]. Specifically, the 1130 sequences in this dataset belong to 37 microbial “genera” or lower-level taxonomic class and had sequence lengths varying from 1330 to 1370 nucleotides (characters). In this paper, we refer to this dataset as DS1.

The second dataset contains 112,283 near-full-length 16S rRNA sequences sampled across 21 different skin locations, from ten healthy human controls. These skin sites were selected because they show a predisposition for bacterial infections. The skin sites are categorized as: (i) sebaceous or oily, (ii) moist (typically skin creases) and (iii) dry, flat surfaces. As given by Grice et. al. [15], the sebaceous sites include locations like glabella (between the eyebrows), alar crease (side of the nostril), external auditory canal (inside the ear), retroauricular crease (behind the ear), occiput (back of the scalp), manubrium (upper chest) and back. Moist sites include the nare (inside the nostril), auxiliary vault (armpit), antecubital fossa (inner elbow), interdigital web space (between the middle and ring fingers), inguinal crease (side of the groin), gluteal crease (top-most part of the fold between the buttocks), popliteal fossa (behind the knee), plantar heel (bottom of the heel of the foot), toe web space and umbilicus (navel). Dry sites include the volar forearm (inside of the mid-forearm), hypothenar palm (palm of the hand proximal to the little finger) and buttock. We use the information about skin locations and “genera” as ground truth in our study. The sequence lengths varied from 1280 to 1370 nucleotides (1300 average). We refer to this dataset as DS2.

4.1.1 Equal Length Datasets. To evaluate the performance of different clustering algorithms on equal length datasets, we perform a pre-processing step where the sequences in DS1 and DS2 are trimmed to the shortest length sequence. This trimming is performed by comparing every sequence to the shortest sequence in the dataset to find the maximum matching gapless sub-sequence of the shortest length. Note, the use of equal length dataset is to understand the characteristics of our algorithm. Our approach works equally well on unequal length sequence datasets.

4.2 Performance Metrics. We evaluate the performance of our clustering algorithm by computing metrics related to computational complexity, accuracy and correctness of the results. For the different parameter settings, we report the run-time and memory used by the MC-LSH algorithm, and compare our performance to UCLUST [4], CD-HIT [3] and CROP [5]. Using ground truth “genera” i.e., taxonomic class labels for each sequence, we determine a weighted average accuracy. Each cluster is assigned to the class/genera which is the most frequent in the cluster, and then the accuracy of this assignment is evaluated by computing the percent of correctly assigned sequences. The reported accuracy is averaged across all clusters, weighted by the

Table 1: Performance Measures of MC-LSH on DS1 and DS2 with Equal Length Sequences.

Iterations (l) = 5 with 10% Mismatch										
Dataset 1						Dataset 2				
Sampled Indices(k) = 64						Sampled Indices(k) = 64				
	# Clu	S.Id	W.Acc	Time	Mem	# Clu	S.Id	W.Acc	Time	Mem
w=1	34	98.36	99.20	2.10	3.0	204	78.65	66.23	347	5.1
w=3	53	98.70	99.40	2.80	4.2	218	79.18	67.38	595	5.8
w=5	59	98.81	100.00	3.30	4.2	223	80.15	67.92	960	6.2
w=7	69	98.82	100.00	3.80	5.5	242	82.18	70.69	1154	6.9
Sampled Indices(k) = 128						Sampled Indices(k) = 128				
w=1	37	98.42	99.30	3.90	3.2	208	78.64	66.72	461	5.1
w=3	54	98.73	99.90	5.10	4.3	227	80.35	68.05	866	6.2
w=5	65	98.83	100.00	6.20	4.3	240	82.29	70.68	1047	6.9
w=7	69	98.82	100.00	7.10	5.5	261	82.75	71.20	2338	7.1
Sampled Indices(k) = 256						Sampled Indices(k) = 256				
w=1	46	98.50	99.80	7.63	3.5	259	82.41	71.96	514	7.1
w=3	62	98.80	100.00	10.38	4.3	287	83.27	72.64	908	8.7
w=5	68	98.82	100.00	11.94	4.4	302	83.90	72.96	2448	9.6
w=7	74	99.08	100.00	13.67	5.6	315	85.45	73.10	3821	9.9

The numbers in bold indicate the parameter settings used to compare the MC-LSH algorithm with UCLUST, CD-HIT and CROP. These parameter settings produce similar number of clusters as produced by other methods and are selected in order to make a fair comparison, though there are many other parameter settings which perform much better than other methods. Number of clusters (# Clu), weighted sequence similarity (S.Id), weighted cluster accuracy in % (W.Acc), MC-LSH Running Time in seconds (Time) and Memory usage in Megabytes (Mem) are the performance metrics.

number of sequences in each cluster. This is denoted by “W.Acc” in this paper.

We also determine for each clustering solution, the sequence similarity within the clusters. Generally, sequence similarity is evaluated by finding the pairwise alignments (i.e., best arrangement of DNA nucleotides (characters) to identify regions of similarity between sequences). Global alignments attempt to align every character between the sequences, and local alignments find the best sub-regions of similar characters [25]. A good sequence clustering solution should produce sequences within a cluster that are similar to each other. We calculate the average global and local sequence alignment similarity across the clusters. Based on similar trends observed for both the local and global sequence alignments, we choose to report only the average global sequence alignment similarity (weighted by number of sequences in a cluster). This quantity is referred to as “S.Id” in this paper.

4.3 Hardware and Software Specifics. The MC-LSH software was written in Matlab version R2007a. All the experiments and simulations were performed on a single workstation, with Intel-i5 2.53 GHz processor and 6 GB memory. Comparative approaches including UCLUST [4], CD-HIT [3] and CROP [5] were also run on the same machine using binary files made available

by the authors of these papers. For CD-HIT and CROP methods, we do not report the memory usage because the binary files are not designed to output this performance measure.

5 Results and Discussion

We perform a comprehensive set of experiments evaluating the performance of MC-LSH algorithm on the two datasets, DS1 and DS2 with equal and unequal length sequences. Our experiments focused on evaluating the performance of MC-LSH algorithm with respect to different parameter settings and also compared our approach to state-of-the-art sequence clustering algorithms.

5.1 Parameter Evaluation. Tables 1 and 2 show the performance of MC-LSH algorithm for both datasets on equal-length and unequal-length sequences, respectively. We performed a series of experiments varying the different parameters of the MC-LSH algorithm but for simplicity we report the results obtained for 5 iterations (l), 10 % mismatch factor, varying w parameter from 1 to 7 and setting the sampled indices parameter k to 64, 128 and 256. Tables 3 and 4 report the results for MC-LSH algorithm by setting the number of sampled bits k to 256, and varying the number of

Table 2: Performance Measures of MC-LSH on DS1 and DS2 with Unequal Length Sequences.

Iterations (l) = 5 with 10% Mismatch										
Dataset 1					Dataset 2					
Sampled Indices(k) = 64					Sampled Indices(k) = 64					
	# Clu	S.Id	W.Acc	Time	Mem	# Clu	S.Id	W.Acc	Time	Mem
w=1	41	98.25	99.04	7.55	3.1	212	78.55	65.12	390	5.3
w=3	59	98.61	99.62	10.78	4.2	223	79.15	66.08	658	6.5
w=5	65	98.70	99.81	16.26	4.3	227	80.02	66.32	1045	7.2
w=7	73	98.72	100.00	21.34	5.5	244	82.10	69.15	1294	7.5
Sampled Indices(k) = 128					Sampled Indices(k) = 128					
w=1	44	98.34	99.08	10.65	3.3	215	78.50	65.14	528	5.4
w=3	61	98.62	99.63	12.70	4.3	233	80.24	67.18	912	6.6
w=5	66	98.72	99.81	21.47	4.3	244	82.17	70.26	1098	7.3
w=7	74	98.79	100.00	27.38	5.6	262	82.63	71.05	2506	7.5
Sampled Indices(k) = 256					Sampled Indices(k) = 256					
w=1	47	98.41	99.14	14.72	3.5	265	82.37	71.24	579	7.2
w=3	63	98.70	99.65	21.39	4.3	290	83.20	72.45	1014	8.9
w=5	69	98.72	99.86	30.62	4.4	304	83.78	72.89	2615	9.8
w=7	78	99.01	100.00	36.95	5.7	316	84.76	73.04	4056	10.1

The numbers in bold indicate the parameter settings used to compare the MC-LSH algorithm with UCLUST, CD-HIT and CROP. These parameter settings produce similar number of clusters as produced by other methods and are selected in order to make a fair comparison, though there are many other parameter settings which perform much better than other methods. Number of clusters (# Clu), weighted sequence similarity (S.Id), weighted cluster accuracy in % (W.Acc), MC-LSH Running Time in seconds (Time) and Memory usage in Megabytes (Mem) are the performance metrics.

iterations ($l = \{1, 2, 5, 10\}$) and the mismatch factor ($r = \{0\%, 5\%, 10\%\}$) for datasets DS1 and DS2, respectively. We report the number of clusters (# Clu), weighted accuracy (W.Acc), weighted sequence identity (S.Id), run-time (Time) and memory used (Mem).

Dataset DS2 in comparison to DS1 (as discussed in Section 4.1) is a larger and more complex dataset. We can see that for DS1, MC-LSH algorithm produces 100% accurate clusters. If we see the results in Table 3, any number of clusters greater than 46 gives 100% accuracy. On the other hand, if we observe the results for large and complex dataset D2, different algorithms do not achieve 100% accuracy.

Effect of varying sampled indices (k): From Tables 1 and 2 we observe that (keeping the other parameters fixed), as the number of sampled indices increase, the number of possible hash values increase, resulting in larger number of clusters. It is more difficult to identify a pair of similar sequences (few mismatches), using the larger sampled indices than smaller sampled indices as the probability of getting the mismatch becomes higher. It is always desirable to achieve low number of clusters with high weighted accuracy. Further, there is a linear relationship with the run-time and sampled indices. The implementation of LSH-based Hamming distance function is computed across the k indices and as we increase the value of k ,

the run-time increases as well.

Another factor that determines the range of sampled indices is the length of sequence and similarity between the sequences. Shorter sequences can be approximately compared with small number of sample indices, whereas larger sequences require large number of sampled indices to capture the patterns. In our experiments, the length of shortest sequence is 1300 and we can see that sampled indices of size 256 is large enough to capture similar patterns among the sequences.

Effect of varying w -mer size: From Tables 1 and 2 we notice that as we increase the w -mer size, the MC-LSH algorithm requires a larger subsequence to be matched for determining if sequence pairs are similar or not. This leads to an increase in the number of clusters and also improvement in the accuracy of the clusters when evaluated with respect to the ground truth. Also, the sequence identity increases because of increased stringency due to the use of matching gapless subsequences instead of single nucleotides (or characters). Our experimental evaluations demonstrate the setting the w -mer size to 5 will prevent under-estimation or over-estimation of the actual number of clusters.

Effect of varying iterations (l): We can observe the effect of increasing the number of hash functions or iterations (l) on the number of clusters, accuracy and

Table 3: MC-LSH Parameter Evaluation on DS1 with Equal Length Sequences.

		0% Mismatch				5% Mismatch				10% Mismatch			
		# Clu	W.Acc	Time	Mem	# Clu	W.Acc	Time	Mem	# Clu	W.Acc	Time	Mem
$l=1$	$w=1$	529	100.00	9.45	1.6	321	100.00	4.01	1.1	132	100.00	1.33	1.0
	$w=3$	697	100.00	11.26	2.1	418	100.00	9.27	1.9	160	100.00	2.18	1.3
	$w=5$	724	100.00	16.68	2.5	442	100.00	10.35	2.1	173	100.00	3.79	1.5
	$w=7$	855	100.00	20.71	3.1	516	100.00	12.26	2.7	198	100.00	4.26	1.7
$l=2$	$w=1$	390	100.00	15.36	2.3	225	100.00	6.72	1.8	73	100.00	2.68	1.8
	$w=3$	522	100.00	21.24	3.2	293	100.00	14.39	2.6	81	100.00	4.17	2.1
	$w=5$	618	100.00	29.08	3.8	347	100.00	18.25	3.5	92	100.00	6.24	2.3
	$w=7$	650	100.00	36.15	4.7	371	100.00	21.78	3.8	105	100.00	8.05	2.9
$l=5$	$w=1$	237	100.00	26.85	4.5	132	100.00	16.8	3.9	46	99.80	7.63	3.5
	$w=3$	433	100.00	59.79	6.1	240	100.00	34.0	5.4	62	100.00	10.38	4.3
	$w=5$	584	100.00	86.71	7.4	315	100.00	48.1	6.8	68	100.00	11.94	4.4
	$w=7$	637	100.00	101.68	8.9	347	100.00	55.6	7.6	74	100.00	13.67	5.6
$l=10$	$w=1$	216	100.00	39.89	9.5	118	100.00	25.3	9.1	37	99.23	13.82	7.0
	$w=3$	343	100.00	96.19	13.8	192	100.00	56.8	10.5	59	100.00	20.11	7.9
	$w=5$	493	100.00	158.36	16.3	269	100.00	87.3	12.1	64	100.00	22.37	8.0
	$w=7$	619	100.00	203.54	18.0	330	100.00	108.7	13.4	72	100.00	25.57	8.1

All experiments are carried out with a fixed Sampled Indices(k) of 256 in order to investigate the maximum utilization and computation of MC-LSH algorithm. Number of clusters (# Clu), weighted cluster accuracy in % (W.Acc), MC-LSH Running Time in seconds (Time) and Memory usage in Megabytes (Mem) are the performance metrics.

computational complexity by analyzing Tables 3 and 4. Increasing the value of l results in merging of clusters and increases individual cluster sizes with each iteration. It is interesting to observe that rate of reduction in the number of clusters is higher when increasing iterations from $l = 1$ to $l = 2$, than varying l from 2 to 5 or 5 to 10. This parameter also affects the accuracy of the algorithm because more dissimilar sequences fall into the same cluster after union operation. Large values of l can lead to false positives, and decrease the accuracy of clusters.

Keeping the other parameters fixed, we can also observe the increase in running time (Time) and memory usage (Mem) for increasing values of l . For example, for the DS2 dataset with equal lengths, setting $w = 1$, $r = 0\%$, and $k = 256$, we see that running time is 293, 498, 852 and 958 seconds for $l = 1, 2, 5$, and 10, respectively.

Effect of varying mismatch factor (r). Analyzing Tables 3 and 4, we see the performance of MC-LSH algorithm for 0%, 5% and 10% mismatch factor across the DS1 and DS2 datasets, respectively.

Increasing the mismatch factor r , leads to a reduction in the number of clusters as well as the accuracy and pairwise similarity within the clusters. This is expected as the mismatch factor is increased, sequences that are dissimilar will satisfy the filter defined by the hamming distance function (Step 10 in Algorithm 1), and hence

will be assigned to the same cluster. A mismatch factor, r of 0% finds the exact k nucleotides or w -mers at the randomly selected positions. Note that the run-time and memory required reduces as we increase the allowable mismatches. The MC-LSH algorithm requires less number of steps to cluster the input sequences (See Algorithm 1), because once a cluster representative is selected, it will find more sequences in the input set within the same cluster, if the allowable number of mismatches is larger.

Equal versus Unequal Length Datasets. Comparing the results of MC-LSH for the equal and unequal length datasets (Tables 1 and 2), we notice that the clusters produced vary slightly in terms of the accuracy and number of clusters. The running time increases for the unequal length datasets in comparison to the equal length datasets due to the increase in computation required for comparing a pair of sequences. For both the datasets, DS1 and DS2 we notice that the difference in run times between the equal and unequal datasets for a larger w -mer size ($w = 7$) is greater than for a smaller w -mer size. Increasing the w -mer size lowers the probability of matching a pair of sequences, and for unequal length sequences this leads to more comparisons between the minimum length subsequence pairs. A similar trend is observed by increasing the number of indices i.e., the parameter k .

Table 4: MC-LSH Parameter Evaluation on DS2 with Equal Length Sequences.

		0% Mismatch				5% Mismatch				10% Mismatch			
		# Clu	W.Acc	Time	Mem	# Clu	W.Acc	Time	Mem	# Clu	W.Acc	Time	Mem
$l=1$	$w=1$	1124	85.12	293	5.0	928	83.96	207	4.8	720	81.62	156	2.0
	$w=3$	1477	89.66	620	6.1	1163	85.75	376	5.5	862	82.76	273	2.2
	$w=5$	1703	91.45	954	7.8	1320	87.29	952	6.2	1067	84.7	745	2.8
	$w=7$	1807	92.18	1305	9.3	1418	89.16	1208	6.9	1125	85.12	1108	3.5
$l=2$	$w=1$	818	81.56	498	8.2	608	79.46	348	6.2	410	73.16	235	3.2
	$w=3$	1075	84.13	1126	10.1	776	81.68	765	7.5	466	76.28	449	4.7
	$w=5$	1240	87.64	1554	13.5	865	82.87	1644	9.1	572	79.02	1276	4.8
	$w=7$	1316	88.25	2169	16.6	933	83.05	2076	11.3	634	80.95	1945	5.1
$l=5$	$w=1$	690	81.27	852	16.7	472	76.90	680	12.1	259	70.96	514	7.1
	$w=3$	907	83.42	2265	21.8	595	79.26	1582	15.3	287	71.45	908	8.7
	$w=5$	1046	84.56	3424	25.8	671	80.92	2933	17.8	302	71.82	2448	9.6
	$w=7$	1110	84.91	4280	31.4	709	81.37	4048	20.7	315	72.04	3821	9.9
$l=10$	$w=1$	651	80.75	948	28.2	440	75.35	786	20.1	239	70.18	628	12.2
	$w=3$	844	82.66	2766	42.1	563	78.58	1941	29.8	284	71.45	1120	16.3
	$w=5$	976	83.52	4057	47.3	617	80.64	3409	32.6	298	71.82	2766	18.4
	$w=7$	1005	83.94	5410	60.8	635	81.05	4798	40.2	309	72.01	4195	19.7

All experiments are carried out with a fixed Sampled Indices(k) of 256 in order to investigate the maximum utilization and computation of MC-LSH algorithm. Number of clusters (# Clu), weighted cluster accuracy in % (W.Acc), MC-LSH Running Time in seconds (Time) and Memory usage in Megabytes (Mem) are the performance metrics.

5.2 Comparative Performance Table 5 shows the comparative analysis of MC-LSH algorithm with UCLUST, CD-HIT and CROP on equal length sequences for datasets DS1 and DS2. The MC-LSH algorithm identifies 37 clusters with weighted cluster accuracy of 99.30%, which is identical to 37 genera taken as ground truth. For DS1, the best clustering result is obtained with the parameter setting $w=1$, $l=5$, $k=128$ with 10% mismatch where MC-LSH algorithm identified 37 genera with weighted cluster accuracy of 99.30% which is similar to 37 genera taken as ground truth. The difference between our results and ground truth is because the genus *Mycobacterium* and genus *Corynebacterium* are represented by two different small clusters. Also, three sequences of genus *Williamsia* is assigned to genus *Propionibacterium* and five sequences of genus *Mycobacterium* fall into genus *Corynebacterium* cluster.

In comparison, UCLUST and CD-HIT identify 30 clusters with 98.29% and 97.79% weighted cluster accuracy respectively, at 90% identity (10% dissimilarity). CROP identifies 33 genera with 98.36% weighted cluster accuracy at 95% identity and also underestimates the number of genera.

For DS2, UCLUST produces 243 and 560 clusters at 90% and 95% identity with 67.14% and 75.25% weighted cluster accuracy respectively. CD-HIT produces 250 and 540 cluster at 90% and 95% identity with 66.76% and 73.4% weighted cluster accuracy, respectively. In

comparison, the MC-LSH results (Table 5) for parameter setting of $w=5$, $l=5$, $k=128$ and $r=10\%$ produces 240 clusters with 70.26% weighted cluster accuracy. With parameters, $w=1$, $l=5$, $k=64$ and $r=0\%$, the MC-LSH algorithm produces 560 clusters with 78.66% weighted cluster accuracy. These parameter settings were chosen because they represent equivalent number of clusters produced by UCLUST and CD-HIT. A similar set of comparative results were presented in Table 6 were presented for unequal length datasets.

The results show that CD-HIT and CROP are time intensive algorithms. The run-time for CROP grows exponentially with increasing dataset size, and crashes for dataset, DS2. Both, CROP and CD-HIT do not provide memory information. MC-LSH is faster than CD-HIT and CROP but slower than UCLUST. The results show that MC-LSH is better than UCLUST, when evaluated with regards to the memory usage. We also observe that in comparison to UCLUST, MC-LSH produces smaller number of clusters with a higher weighted accuracy. The results for the quality of clusters evaluated using sequence identity (global as well as local) are fairly similar for all approaches.

6 Significance and Impact.

Metagenomic projects aim to determine the species similarity and diversity across different ecological samples. We use the MC-LSH algorithm to provide an enriched

Table 5: Comparative Performance on DS1 and DS2 with Equal Length Sequences.

		Dataset 1				Dataset 2			
		# Clu	W.Acc	Time	Mem	# Clu	W.Acc	Time	Mem
MC-LSH	w=1,l=10,k=256,10%	37	99.20	13.80	7.0	239	70.25	628	12.2
	w=5,l=5,k=128,10%	65	100.00	6.20	4.3	240	70.26	1047	6.9
	w=7,l=5,k=128,5%	102	100.00	7.10	5.5	316	72.04	2510	9.5
	w=1,l=5,k=64,0%	135	100.00	6.90	7.5	560	78.66	524	10.8
UCLUST	Identity 100%	1065	100.00	1.10	30.0	75577	98.50	130	1100.0
	Identity 95%	37	98.14	0.65	8.2	560	75.25	62	34.0
	Identity 90%	29	98.02	0.50	8.1	243	67.14	35	28.0
CD-HIT	Identity 100%	1005	100.00	8.20	-	63336	98.05	52115	-
	Identity 95%	39	98.04	18.25	-	540	73.40	10706	-
	Identity 90%	29	97.65	20.50	-	250	66.76	7545	-
CROP	Identity 97%	41	98.70	9562	-	-	-	-	-
	Identity 95%	32	98.25	10383	-	-	-	-	-

UCLUST, CD-HIT and CROP use a percentage identity parameter which ensures that all pairs of sequences in a cluster must have at least that percentage of sequence similarity. For MC-LSH, w is the w -mer, l is number of hash function or iterations, k is the Sampled Indices and mismatch is the percentage of dissimilar w -mers allowed. Number of clusters (# Clu), weighted cluster accuracy in % (W.Acc), MC-LSH Running Time in seconds (Time) and Memory usage in Megabytes (Mem) are the performance metrics. The numbers in bold highlight the better performance of MC-LSH algorithm over other methods. CD-HIT does not provide memory usage in the results and CROP is unable to give any results on large dataset DS2.

analysis of the variation of microbial species across different skin locations (a real environmental dataset). Clustering of metagenomic sequences (16S), tends to group sequences that are closely related in the taxonomy/phylogenetic tree (genera) together.

Using the MC-LSH algorithm, we cluster all the metagenomic sequences available in DS2 into 244 clusters. Sequences in DS2 are from 21 different skin locations (Section 4.1) and are categorized based on their property into three classes: (i) moist, (ii) dry and (iii) sebaceous (oily). Each skin location is a metagenomic sample i.e., genomic sequences of microbes co-existing on the specific skin locations.

After clustering the sequences across all the locations, each sequence is associated with one of the cluster labels. Using the cluster memberships as features for a skin location, we compute the Jaccard index between all pairs of skin locations. Jaccard index measures the proportion of shared clustering labels (species) between the pair of skin locations. A higher index value indicates that the two skin locations are more similar to each other.

Figure 1 shows the pairwise relationship between the different skin locations as measured by Jaccard index. We also show the hierarchical clustering produced by using the similarity measure. We can observe that some skin locations which are considered to be similar according to the cluster-membership are not similar in terms of their physical property. For example, the *antecubital fossa* is similar to *volar forearm* according to the Jaccard index but *antecubital fossa* is a moist

skin location whereas the *volar forearm* is a dry skin location. An interesting fact about these skin locations is that they both are on the “human forearm”. This suggests that the microbial content across the skin is determined by the physical property (e.g., dry or moist) and the spatial location (e.g., forearms). This hypothesis was verified by an independent skin microbiome study by Costello et. al. [26], and the hierarchical clustering produced in Figure 1 was validated by the CROP algorithm.

The results demonstrate the strength of MC-LSH in producing meaningful results and use within a larger metagenome analysis pipeline. In Figure 2, we observe that MC-LSH produces accurate and large clusters. MC-LSH can also be used for reducing the complexity associated with analyzing large number of sequences in a metagenomic/genomic samples. This could be done by performing downstream analysis only on the cluster representatives rather than individual sequences.

7 Conclusion

In this paper, we present a new, scalable metagenomic sequence clustering algorithm (MC-LSH) that utilizes an efficient locality sensitive based hashing function to approximate the pairwise sequence operations. The basic LSH-function was enriched to use gapless, subsequences of fixed length (w -mer), which lead to a reduction in the number of false positives and improvement of cluster accuracy. We evaluated MC-LSH on two metagenomic datasets and performed a comprehen-

Table 6: Comparative Performance on DS1 and DS2 with Unequal Length Sequences.

		Dataset 1				Dataset 2			
		# Clu	W.Acc	Time	Mem	# Clu	W.Acc	Time	Mem
MC-LSH	w=1,l=10,k=256,10%	37	99.10	20.86	7.1	241	70.18	662	12.5
	w=5,l=5,k=128,10%	66	99.81	21.47	4.3	244	70.19	1098	7.3
	w=7,l=5,k=128,5%	108	100.00	27.38	5.7	323	71.34	2766	9.7
	w=1,l=5,k=64,0%	141	100.00	13.85	7.7	563	77.80	542	11.0
UCLUST	Identity 100%	1068	100.00	1.30	30.0	75589	98.56	169	1102.0
	Identity 95%	39	98.37	0.80	8.2	571	75.32	74	35.0
	Identity 90%	30	98.29	0.60	8.1	248	67.55	43	29.0
CD-HIT	Identity 100%	1009	100.00	9.10	-	63342	98.18	54178	-
	Identity 95%	40	98.16	19.55	-	549	73.52	10855	-
	Identity 90%	30	97.79	22.00	-	257	66.87	7632	-
CROP	Identity 97%	43	98.82	9580	-	-	-	-	-
	Identity 95%	33	98.36	10415	-	-	-	-	-

UCLUST, CD-HIT and CROP use a percentage identity parameter which ensures that all pairs of sequences in a cluster must have at least that percentage of sequence similarity. For MC-LSH, w is the w -mer, l is number of hash function or iterations, k is the Sampled Indices and mismatch is the percentage of dissimilar w -mers allowed. Number of clusters (# Clu), weighted cluster accuracy in % (W.Acc), MC-LSH Running Time in seconds (Time) and Memory usage in Megabytes (Mem) are the performance metrics. The numbers in bold highlight the better performance of MC-LSH algorithm over other methods. CD-HIT does not provide memory usage in the results and CROP is unable to give any results on large dataset DS2.

sive study of the different parameters and their impact on run time, memory usage and cluster accuracy. We demonstrated that MC-LSH was computationally efficient in comparison to the state-of-the-art clustering algorithms, and also produced meaningful and accurate clustering results with respect to external class labels. We also showed a use case scenario for metagenomics clustering to evaluate the species diversity and dynamics across 21 different skin locations.

As part of our future work we are developing an analytical method for determining the right combination of parameters. This is based on assuming the false positive and false negative rates that could be expected, and analyzing the distribution of sequence lengths as well as pairwise sequence identity within metagenomic samples. Donga et. al [27] have proposed an approach for performance tuning of LSH. The algorithm can be easily parallelized, because the different operations related to computing the distance function offer concurrency. We plan to develop a parallel version of MC-LSH that can utilize distributed computing resources.

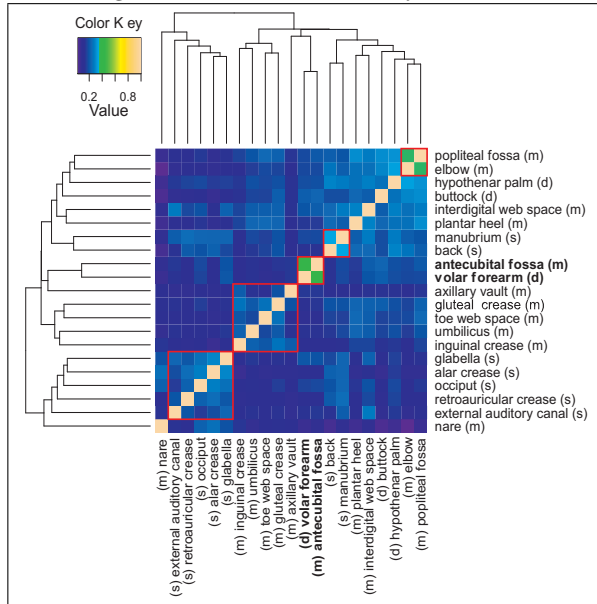
Acknowledgment

The work is supported by NSF grant IIS 0905117 and bio engineering seed grant awarded to HR.

References

- [1] S. G. Tringe and et. al. Comparative metagenomics of microbial communities. 308(5721):554–7+, 2005.
- [2] Gene W. Tyson and Philip Hugenholtz. Metagenomics. *Nature Reviews Microbiology*, Sep 2008.
- [3] Weizhong Li and Adam Godzik. CD-HIT: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659, 2006.
- [4] Robert C. Edgar. Search and clustering orders of magnitude faster than blast. *Bioinformatics*, 2010.
- [5] Xiaolin Hao, Rui Jiang, and Ting Chen. Clustering 16s rRNA for OTU prediction: a method of unsupervised bayesian clustering. *Bioinformatics*, 27(5):611–618, 2011.
- [6] Patrick D. Schloss and Jo Handelsman. Introducing dotur, a computer program for defining operational taxonomic units and estimating species richness. *Appl. Environ. Microbiol.*, 71(3):1501–1506, 2005.
- [7] D Schloss, Patrick and et. al. Introducing mothur: Open-source, platform-independent, community-supported software for describing and comparing microbial communities. *Appl. Environ. Microbiol.*, 75(23):7537–7541, 2009.
- [8] Yijun Sun and et. al. Esprit: estimating species richness using large collections of 16s rRNA pyrosequences. *Nucleic Acids Research*, 2009.
- [9] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM.
- [10] Yiqun Cao, Tao Jiang, and Thomas Girke. Accelerated similarity searching and clustering of large compound sets by geometric embedding and locality sensitive hashing. *Bioinformatics*, 26(7):953–959, 2010.
- [11] Christopher Kruegel and et. al. Scalable, behavior-based malware clustering. In *Proceedings of the 16th*

Figure 1: Pairwise similarity for DS2.

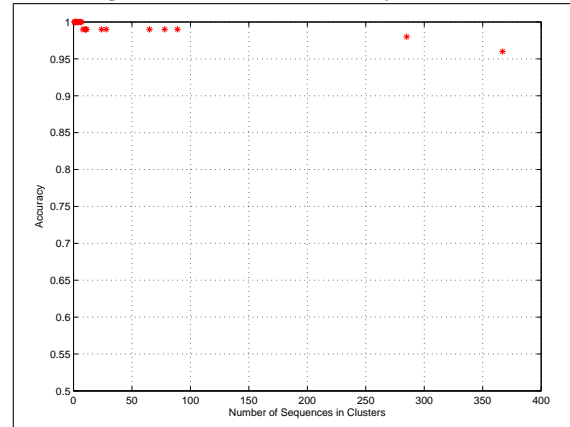


Similarity between different skin location and types using Jaccard Index can be visualized by color key and hierarchical clustering plot. (m), (s) and (d) are defined as moist, sebaceous, and dry, respectively. The parameter setting for MC-LSH was $w=5$, $l=5$ and $k=128$ with 10% mismatch.

Annual Network and Distributed System Security Symposium (NDSS 2009), 1 2009.

- [12] Hisashi Koga, Tetsuo Ishibashi, and Toshinori Watanabe. Fast agglomerative hierarchical clustering algorithm using locality-sensitive hashing. *Knowl. Inf. Syst.*, 12:25–53, May 2007.
- [13] Jeremy Buhler. Efficient large-scale sequence comparison by locality-sensitive hashing. *Bioinformatics*, 17(5):419–428, 2001.
- [14] Susan Huse, Julie Huber, Hilary Morrison, Mitchell Sogin, and David Welch. Accuracy and quality of massively parallel dna pyrosequencing. *Genome Biology*, 8(7):R143+, jul 2007.
- [15] Elizabeth A. Grice and et. al. Topographical and temporal diversity of the human skin microbiome. *Science*, 324(5931):1190–1192, 2009.
- [16] J.C. Venter and et. al. Environmental genome shotgun sequencing of the Sargasso Sea. *Science*, 304(5667):66, 2004.
- [17] Junjie Qin et al. A human gut microbial gene catalogue established by metagenomic sequencing. *Nature*, 464(7285):59–65, Mar 2010.
- [18] Peter J Turnbaugh and et. al. The human microbiome project. *Nature*, 449(7164):804810, October 2007.
- [19] G Muyzer, E C de Waal, and A G Uitterlinden. Profiling of complex microbial populations by denaturing gradient gel electrophoresis analysis of polymerase chain reaction-amplified genes coding for 16s rRNA. *Appl. Environ. Microbiol.*, 59(3):695–700, 1993.

Figure 2: Cluster Accuracy for DS1.



The figures shows a plot of accuracy per cluster and the number of sequences within the cluster. MC-LSH shows high accuracy for large clusters. Parameters: $w=1$, $l=10$, $k=256$ and 10% mismatch produces 37 clusters with 99.20% W.Acc.

- [20] Soumitesh Chakravorty, Danica Helb, Michele Burday, Nancy Connell, and David Alland. A detailed analysis of 16s ribosomal rna gene segments for the diagnosis of pathogenic bacteria. *Journal of Microbiological Methods*, 69(2):330 – 339, 2007.
- [21] Zeesham Rasheed and Huzefa Rangwala. TAC-ELM: Metagenomic taxonomic classification with extreme learning machines. *Third International Conference on Bioinformatics and Computational Biology (BICoB-2011)*, March 2011.
- [22] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases, VLDB '99*, pages 518–529, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.
- [23] P.A. Pevzner, H. Tang, and M.S. Waterman. An Eulerian path approach to DNA fragment assembly. *Proceedings of the National Academy of Sciences of the United States of America*, 98(17):9748, 2001.
- [24] Huzefa Rangwala and George Karypis. Profile-based direct kernels for remote homology detection and fold recognition. *Bioinformatics*, 21(23):4239–4247, Dec 2005.
- [25] Xiaoqui Huang. On global sequence alignment. *Computer applications in the biosciences : CABIOS*, 10(3):227–235, 1994.
- [26] Elizabeth K. Costello and et. al. Bacterial community variation in human body habitats across space and time. *Science*, 326(5960):1694–1697, 2009.
- [27] Wei Dong and et. al. Modeling lsh for performance tuning. In *Proceedings of the 17th ACM conference on Information and knowledge management, CIKM '08*, pages 669–678, New York, NY, USA, 2008. ACM.