HMSC: a Hybrid Metagenomic Sequence Classification Algorithm

Subrata Saha
Healthcare and Life Sciences Division
IBM Research
Yorktown Heights, New York, U.S.A
s.saha@ibm.com

Zigeng Wang
Dept. of Computer Science & Engr.
University of Connecticut
Storrs, Connecticut, U.S.A
zigeng.wang@uconn.edu

Sanguthevar Rajasekaran*
Dept. of Computer Science & Engr.
University of Connecticut
Storrs, Connecticut, U.S.A
sanguthevar.rajasekaran@uconn.edu

ABSTRACT

Widespread availability of next-generation sequencing (NGS) technologies has prompted a recent surge in interest in the microbiome. As a consequence, metagenomics is a fast growing field in bioinformatics and computational biology. An important problem in analyzing metagenomic sequenced data is to identify the microbes present in the sample and figure out their relative abundances. Genome databases such as *RefSeq* and *GenBank* provide a growing resource to characterize metagenomic sequenced datasets. However, both the size of these databases and the high degree of sequence homology that can exist between related genomes mean that accurate analysis of metagenomic reads is computationally challenging. In this article we propose a highly efficient algorithm dubbed as "Hybrid Metagenomic Sequence Classifier" (HMSC) to accurately detect microbes and their relative abundances in a metagenomic sample. The algorithmic approach is fundamentally different from other state-of-the-art algorithms currently existing in this domain. HMSC judiciously exploits both alignment-free and alignment-based approaches to accurately characterize metagenomic sequenced data. Rigorous experimental evaluations on both real and synthetic datasets show that HMSC is indeed an effective, scalable, and efficient algorithm compared to the other state-of-the-art methods in terms of accuracy, memory, and runtime.

KEYWORDS

Hybrid Metagenomic Sequence Classifier (HMSC); Metagenomics; CLARK; Kraken

ACM Reference Format:
Subrata Saha, Zigeng Wang, and Sanguthevar Rajasekaran. 2020. HMSC: a Hybrid Metagenomic Sequence Classification Algorithm. In *Proceedings of the 11th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics (BCB '20), September 21–24, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3388440.3412468

*Sanguthevar Rajasekaran is the corresponding author.

BCB'20, September 21–24, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7964-9/20/09…$15.00
https://doi.org/10.1145/3388440.3412468

1 INTRODUCTION

Although we are normally unable to see microbes, they run the world. Microbes are indispensable for every part of our human life - truly speaking - all life on Earth! They influence our daily life in a myriad ways. For an example, the microbes living in the human gut and mouth enable us to extract energy from food. We could not be able to digest food without them. Microbes also insulate us against disease-causing agents. Metagenomics is a strong tool that can be used to decipher microbial communities by directly sampling genetic materials from their natural habitats. It can be directly applicable a wide variety of domains to solve practical challenges such as biofuels, food safety, agriculture, medications, etc. Metagenomic sequencing shows promise as a sensitive and rapid method to diagnose infectious diseases by comparing genetic material found in a patient's sample to a database of thousands of bacteria, viruses, and other pathogens. There is also strong evidence that microbes may contribute to many non–infectious chronic diseases such as some forms of cancer, coronary heart disease, and diabetes.

Classifying metagenomic sequences in the metagenomic sample is a very challenging task due to these facts: (1) researchers have sequenced complete genomes of thousands of microbes. The total size of the genomes is hundreds of gigabytes. Again, metagenomic sample can contain millions to billions of biological sequencing reads and their total size can range from gigabytes to terabytes. To detect the microbes, we must align the metagenomic reads onto the known reference genomes. Processing is difficult to accomplish as we have time and memory bound; and (2) different microbes can contain similar genomic regions in their genomes. Reads in the given metagenomic sample may refer to different microbes that are not present in the sample. Consequently, there might be a lot of false identification if we just align the reads onto the references. There are several subsequence-based approaches in this domain. These methods suffer from low classification accuracy, high execution time, and high memory usage. The users also need to post-process the output. Thus, we need efficient and effective computational technique to quickly and accurately identify microbes present in metagenomic sample. To address the issues stated above we have developed a highly efficient method to correctly identify and estimate the abundances of microbes in the metagenomic sample.

2 RELATED WORKS

The traditional approach to solve the metagenomic sequence classification is to align each input read onto a large collection of reference genomes using alignment software, such as BLAST [3] or MegaBlast [15]. However, aligning the reads onto the reference

sequences requires a huge computing time. One way of salvaging execution time could be to align the reads onto a select marker genes in the reference genome instead of the whole genome. This approach is followed in Metaphan [23], Metaphyler [11], Motu [21] and Megan [8]. However, this approach also becomes infeasible when there are more and more of reference genomes and the total number of reads grows more and more.

Researchers have tried several alignment free methods. The most popular among such methods is based on k-mer spectrum analysis which uses a database of distinct subsequences of length k, or in short k-mers, from the reference sequence to classify the reads. If a read has distinct k-mers from multiple reference genomes then it is assumed to be from their lowest common taxonomic ancestor (see, e.g. [24]). The algorithms using this broad approach differ in the way the database is built and queried. LMAT [1] is one of the first such algorithms. Subsequent improvements are: Kraken [24] which improves the speed and memory usage by employing a classification tree. CLARK [18] improves memory usage further by storing only a reduced set of target specific k-mers. CLARK-S [17] improves the specificity of CLARK by sacrificing a little speed and memory. Metacache [16] improves the memory usage further by a novel application of minhashing technique on a subset of the context aware k-mers to reduce memory usage. MetaOthello [12] uses a probabilistic hashing classifier and improves the memory usage and k-mer query efficiency with a novel I-Othello data structure. LiveKraken [22] classifies reads in realtime from raw data streams from Illumina sequencers. KrakenUniq [2] efficiently assesses the coverage of unique k-mers and gives better recall and precision.

Kaiju [14] uses the same k-mer based approach, however, exploits the fact that microbial and viral genomes are typically densely packed with protein-coding genes which are more conserved and more tolerant to sequencing errors because of the degeneracy of the genetic code. Some applications require an estimate of the relative abundance of constituent species. Using a probabilistic method based on Byesian likelihood, Braken [13] augments the output of Kraken with estimated abundance. There have been few alignment free methods other than the k-mer spectrum analysis as well. MetaKallisto [20] uses pseudo alignments. WGSQuikr [10] and Metapallette [9] use a compressed sensing approach where the abundance is estimated by solving a linear system of equations on the k-mer spectrum profile of input reads and that of the reference sequences. TaxMap [4] uses a compression algorithm to store the LCA information and achieves better precision and sensitivity.

Recently, deep learning models [6], such as convolutional neural network (CNN) and deep belief network (DBN), have been tested on a small subset of bacteria taxonomic classification, but it is still challenging for the classification for the entire bacteria domain.

3 METHODS

3.1 Our Algorithms

There are 3 major steps in our proposed algorithm HMSC. The first step involves collecting a set of unique k-mers from each of the genomic sequences of interest. This step is different from the k-mer counting problem. In k-mer counting we compile all the distinct k-mers present in the input sequences together with the frequency of each k-mer. A k-mer can be found in more than one genomic sequence. Conversely, each unique k-mer can be found in one and only one genome sequence. The second step involves finding a set of *discriminating regions* from each genome. We then build a model sequence for each genome by adopting the discriminating regions. Instead of using the full genome sequences, these pre-built model sequences are then employed to accurately profile all the 8 taxonomic ranks. We also estimate approximate abundances of all the 8 taxonomic ranks residing in a given metagenomic sample. Experimental evaluations show that HMSC is highly efficient in terms of accuracy, execution time, and memory footprint. Next, we describe the algorithmic steps of HMSC in detail.

3.1.1 ***Unique k-mer mining.*** At the beginning, we identify a set of unique k-mers from the given set of target genomes $G = \{g_1, g_2, g_3, \ldots, g_n\}$. A k-mer is said to be unique if it (and its reverse complement) occurs in only one of the genomes $g_i \in G$ where $1 \leq i \leq n$. However, if we want to search for unique k-mers from the set of all such k-mers in one single pass, it will be a very memory intensive and time consuming procedure. For instance, In our proposed algorithm we employ around $6k$ bacterial genomes and each genome contains nearly $4.5M$ nucleotides on an average! To reduce the memory footprint we partition G into P non-overlapping parts. In each such part p_j (where $1 \leq j \leq P$) we search for k-mers that are unique across all the genomes. To reduce the search space we randomly select a subset of all the k-mers present in sequences.

For each partition $p_j \in P$ we maintain two data structures to find the unique k-mers in each genome. Hash table H is used to record unique k-mers. Each key of the hash table H represents a k-mer and its corresponding value refers to the associated genome id and start index. Hash set S contains the non-unique k-mers found in more than one genome. In both data structures the expected time complexity to search, update, or delete operations is $O(1)$. After collecting locally unique k-mers by utilizing H and S for each genome in one part, we collect the unique set of k-mers from the reduced set of locally unique k-mers found in the previous step by checking if it occurs in the other genomes. We save the unique k-mers picked from each part with the associated genome ids and start indices in the disk. Next we describe the method in detail.

For each genome g in a part p_j, $1 \leq j \leq P$, we process it as follows. We pick a random subset of k-mers in g. Let x' and x'' be any such k-mer and its reverse complement, respectively. Suppose x is the lexicographically smallest of x' and x''. We first check if x is already declared as non-unique by searching for it in the hash set S. If S already contains x, it is non-unique. Otherwise, we search for x in the hash table H. There are two possibilities: there is an entry for x in H or there is no entry for x in H. If there is an entry for x in H, there are two possibilities: (1) The k-mer in the entry corresponds to another genome or (2) It corresponds to the same genome (as that of x). In the former case we declare x as non-unique by recording x in S and delete the entry from H. If the later is the case, we do not do anything. If we do not find any entry for x in H, we create an entry for x in H associated with its genome id and position. At the end of processing all the randomly picked k-mers in all the genomes in p_j, in the above manner, H has all the locally unique k-mers in p_j. Please, note that these locally unique k-mers are only unique with respect to the randomly picked k-mers from the genomes in p_j. From out of these locally unique k-mers,

we identify globally unique k-mers. To find the globally unique k-mers, we iteratively retrieve each genome $g' \in G$ from the disk. To check the duplicity efficiently we build a hash set S' containing the lexicographically smallest k-mers of g'. I.e., for each k-mer in the retrieved genome g', we insert into S' the lexicographically smallest one between it and its reverse complement. Now the locally unique set of k-mers are checked for duplicity against S'. Note that the genomes are retrieved from the disk into the main memory one at a time. The locally unique k-mers will also be checked for duplicity with respect to the genomes in p_j. Once we identify the globally unique k-mers in p_j, we save them together with their positions and genome ids in the disk.

As noted earlier, we partition G into P parts (for some suitable value of P) to reduce the memory footprint. For each of the P parts we follow the same procedure as described above. We save the globally unique k-mers along with their genome ids and positions in the disk for each part. In the experiments we set $k = 31$.

3.1.2 ***Model sequence formation.*** In this step we build a model sequence for each genome $g \in G$. Let the set of unique k-mers in g (found in the previous step) be u. At first we sort u in increasing order with respect to the starting coordinates of the k-mers. Next we cluster the sorted k-mers in such a way that in any cluster the distance between any pair of consecutive k-mers is $\leq$ a threshold t_1. Let a pair of consecutive k-mers be k' and k''. Then the distance between the end position of k' and the start position of k'' will be no more than t_1. We linearly search through the sorted list of k-mers and build a new cluster when the distance between any pair of consecutive k-mers is $> t_1$. Let the set of clusters be C. Clearly, each cluster $c \in C$ contains a set of consecutive k-mers where the distance between any two consecutive k-mers is $\leq t_1$. Consequently, each cluster $c \in C$ contains a discriminating region of the genome g. We extract a region from g by using the start index of the first k-mer and the end index of the last k-mer in c. We call such a region *discriminating* since if any read r having length $\geq t_1 + 2k$ is aligned onto such a region, then r will fully contain at least 1 unique k-mer. In our experiments we set $t_1 = 40$.

We sort all such regions from all the clusters c based on decreasing order of their lengths. We discard some regions having length $\leq t_2$, a user defined threshold. We append a special string of length 4 containing a special character "#" at the end of each region. We concatenate all such regions of a genome g to build a model sequence. Because of this special string no aligned read will contain the junction of any 2 regions given that the mismatch threshold $d < 4$. Let these model sequences are $m_1, m_2, \ldots, m_n$ where m_i is the model sequence of g_i, $1 \leq i \leq n$. In our experiments we set $t_2 = 300$.

3.1.3 ***Taxonomic ranks identification.*** In this step we are inferring all the 8 taxonomic ranks (e.g., subspecies, species, genus, family, order, class, phylum, and kingdom) and their corresponding relative abundances from a given metagenomic sample V. Metagenomic sequencing reads contained in V are aligned onto each of the model sequences m_i built in the previous step. Suppose a read $r \in V$ is aligned onto a model sequence m_i within a certain mismatch threshold d (we set $d = 0$ in our experiments). We can say that the read r belongs to the genome g_i with a high level of confidence. This is referred to as a *hit*.

If a read r is aligned onto multiple model sequences m_i from multiple genomes $g_i \in G$, then the read r is assigned to all of those genomes g_i. Since the taxonomic profiling of HMSC is based on the model sequences of the genomes, not all the reads r from the metagenomic sample V will be classified. This is due to the fact that the model sequences may not contain all the stretches of the original genomes. We estimate the abundance of a specific taxonomic rank using the number of hits. Consider a specific genome g_i. Let the taxonomic rank of g_i be t_i. We estimate t_i by taking the ratio of the hits in m_i with respect to that taxonomic rank to the total number of hits (across all the model sequences).

The accuracy of our algorithm has been measured using the ground truth. For instance, if we employ synthetic data, then we will know what species are represented in the sample V and also the number of reads in V corresponding to each of the species.

4 EXPERIMENT RESULTS

4.1 *Mock* Microbial Metagenomes

A mock community of 36 bacterial species prevalent in the human microbiome [19] was created as described by [5]. Sequencing libraries were created using the *Illumina TruSeq Nano DNA HT kit* and sequenced on the *Illumina HiSeq 2000* platform to generate 2×150 bp paired-end reads. Prior to analysis reads were processed with Trimmomatic to remove sequencing adapters. Following the same procedure we also generated another mock community having 11 bacterial species.

4.2 *In Silico* Microbial Metagenomes

We simulated 6 metagenomic paired-end *in silico* datasets by employing various Illumina platforms. Reads are produced by engaging a shotgun sequence simulator named InSilicoSeq [7]. At first, we randomly selected 200 genomes from around 6k "complet" genomes from *GeneBank*. Please, note that all the randomly selected genomes also appeared in the databases of CLARK and Kraken. We generated simulated paired-end reads D1-D6 containing randomly chosen 50 and 100 bacterial species from the set of 200 genomes as stated above by employing 3 popular Illumina error models (e.g. HiSeq, MiSeq, and NovaSeq) with realistic abundance distribution. The details of the datasets can be found in Table 1.

Table 1: Dataset Information.

Data	Model	Genomes	Paired-end reads	Read length
D1	HiSeq	50	3M	125
D2	HiSeq	100	5M	125
D3	MiSeq	50	3M	300
D4	MiSeq	100	5M	300
D5	NovaSeq	50	3M	150
D6	NovaSeq	100	5M	150
D7	HiSeq	11	4.4M	150
D8	HiSeq	36	6.1M	150

4.3 Performance Metrics

To demonstrate the utility of unique k-mer based model sequences for taxonomic profiling we compared the performance of HMSC with two widely used, k-mer-based tools, CLARK and Kraken, selected for their high accuracy and low execution time. We computed the following four performance metrics to demonstrate the efficacy of our algorithm HMSC:

- ***Recall:*** In information retrieval, *recall* is the fraction of the taxa level (e.g. genus, species, etc.) in the metagenomic sample that are successfully detected. It is defined as: $Recall = \frac{TP}{TP+FN}$. Here TP stands for the number of true positives, i.e. the number of taxa present in the sample and correctly identified by an algorithm. FN stands for the number of false negatives, i.e. the number of taxa present in the sample and not identified by an algorithm.
- ***Precision:*** It is the fraction of retrieved taxa levels that are residing in the sample. The definition is: $Precision = \frac{TP}{TP+FP}$. Here FP is the number of false positives, i.e. the number of taxa identified by an algorithm that are not in the sample.
- ***F measure:*** It is the harmonic mean of *precision* and *recall*, the traditional F-measure or balanced F-score: $F = 2 \times \frac{precision \times recall}{precision + recall}$.
- ***Pearson's Moment Correlation Coefficient (PMCC):*** We use PMCC to measure how much estimated relative abundances of taxonomic ranks are correlated with respect to ground truths. In statistics, the Pearson moment correlation coefficient (PCC), also referred to as Pearson's r, is a measure of the linear correlation between two variables X and Y. We can think of X and Y as 2 vectors each having N entries. The mathematical formulation is: $r = \frac{N \sum XY - (\sum X \sum Y)}{\sqrt{[N \sum x^2 - (\sum x)^2][N \sum y^2 - (\sum y)^2]}}$

We know the unique taxonomic id of each microbe residing in the simulated datasets *a priori*. From a taxonomic id we can retrieve all the taxonomic ranks of a microbe by traversing the taxonomy tree of life. From these ground truths (i.e, taxonomic ids) of all the datasets we identify each taxonomic rank t of all the microbes. Suppose A that belongs to a specific taxonomic rank t (such as, genus). For each algorithm we also identify the same taxonomic rank t predicted, B. We compute *recall* and *precision* as $|A \cap B|/|A|$ and $|A \cap B|/|B|$, respectively. For each algorithm we compute the performance metrics for running each taxonomic profiler on the *mock* and *in silico* communities.

4.4 Precision, Recall, and F Measure

The performance of a classification algorithm depends on both *precision* and *recall*. An algorithm can have a high *recall* but a small *precision* due to the fact that the algorithm with a low *precision* suffers from high false positives. To logically fix the issue the classification performance of an algorithm is measured by taking the harmonic mean of *recall* and *precision* (known as *F measure*). It is observed that the existing algorithms for classifying metagenomic sequences suffer from very high false positives, i.e. they inaccurately identifies a large number of microbes that does not belong to the metagenomic sample.

At first, consider the *in silico* datasets. As noted earlier our *in silico* datasets consist of 6 metagenomic samples (please, see D1-D6 in Table 1). HMSC possesses perfect *recall* of 1.0 for every *in silico* datasets i.e., it was able to identify all the microbes (and their associated taxonomic ranks) prevalent in the samples. Although HMSC detects microbes that are not in the samples (i.e., false positives), the numbers are far smaller than CLARK or Kraken. It is evident from *precision* and *F measure* - HMSC's *precisions* and *F measures* are higher than CLARK and Kraken for all taxonomic ranks in every datasets. Please, note that we only show 6 taxonomic ranks (e.g., subspecies, species, genus, family, order, and class) in Table 2 and 4 taxonomic ranks (e.g., subspecies, species, genus, and family) in Figure 1 because of space constraints.

Now consider mock datasets (D7-D8). In D7 dataset HMSC's *recall* of species-level taxonomy is better than that of CLARK and Kraken. In all other cases *recall* measures are identical for all 3 algorithms. On the contrary *precision* and *F* measures are higher than that of CLARK and Kraken in both of D7 and D8 datasets. It is evident from Table 3 that both of the algorithms erroneously identify a lot of microbes that are not residing in the sample.

4.5 Relative Abundances

Our algorithm HMSC deals with the model sequences of the reference genomes. Each model sequence comprises a set of discriminating regions of a genome. Therefore, all the reads coming from a specific genome will not be aligned onto the model sequence designated for that genome. Only the discriminating reads will be aligned onto a specific model sequence. As model sequences contains discriminating stretches of genome sequences, it estimates approximate relative abundances instead of true relative abundances. Although in theory the approximation may be over-represented or under-represented, HMSC mimics, in practice, the true relative abundances in most of cases. Since we do not know the true relative abundances of the 2 mock communities (e.g., D7-D8), we could not be able to compute Pearson's correlations. Please, see Figure 1[b] for visual comparisons of different methods employed including HMSC. It is to be noted that the abundance estimations of HMSC in MiSeq datasets are poor with respect to CLARK and Kraken. It might be due to the fact that we are aligning reads onto model sequences without any mismatches for every datasets to preserve uniformity. Since the length of MiSeq reads is 300bp long, many of them might not be aligned onto the model sequences within the mismatch threshold we used (i.e., $d = 0$).

4.6 Execution Time and Memory Consumption

HMSC has a very low memory footprint. On average, HMSC uses 3.70× and 1.43× less memory than that of CLARK and Kraken, respectively. The execution times of HMSC are also comparable with the state-of-the-art algorithms. In general, HMSC is faster than CLARK on real datasets D7 and D8. Comparing to Kraken, HMSC's running time is in the same order of magnitude. Please, see Figure 1[c] for visual comparisons.

5 CONCLUSION

In this article we propose HMSC that can accurately detect microbes and their relative abundances in a metagenomic sample. The algorithm judiciously exploits both alignment-free and alignment-based approaches and our rigorous experimental evaluations show that it is indeed an effective, scalable, and efficient algorithm compared to

Table 2: Performance Evaluations on *in silico* Datasets.

		HMSC				CLARK				Kraken			
		Recall	Precision	F-score	PMCC	Recall	Precision	F-score	PMCC	Recall	Precision	F-score	PMCC
D1	Subspecies	1.0000	0.3636	0.5333	0.9924	NA	NA	NA	NA	1.0000	0.0935	0.1709	0.9884
	Species	1.0000	0.2816	0.4395	0.9860	0.9796	0.1627	0.2791	0.9998	0.9796	0.0673	0.1260	0.9998
	Genus	1.0000	0.4737	0.6429	0.9839	1.0000	0.2500	0.4000	1.0000	1.0000	0.1402	0.2459	0.9999
	Family	1.0000	0.7647	0.8667	0.9896	1.0000	0.3391	0.5065	1.0000	1.0000	0.2308	0.3750	1.0000
D2	Subspecies	1.0000	0.4390	0.6102	0.9892	NA	NA	NA	NA	0.9722	0.0508	0.0966	0.9776
	Species	1.0000	0.4091	0.5806	0.9897	0.9697	0.0695	0.1297	0.9835	0.9798	0.0470	0.0897	0.9846
	Genus	1.0000	0.6385	0.7793	0.9895	0.9880	0.1312	0.2316	0.9861	0.9880	0.1004	0.1822	0.9870
	Family	1.0000	0.8571	0.9231	0.9903	1.0000	0.2661	0.4204	0.9979	1.0000	0.2185	0.3587	0.9985
D3	Subspecies	1.0000	0.2899	0.4494	0.5898	NA	NA	NA	NA	1.0000	0.1047	0.1896	0.8758
	Species	1.0000	0.2450	0.3936	0.6536	0.9796	0.0779	0.1444	96.11	0.9796	0.1064	0.1920	0.9670
	Genus	1.0000	0.5056	0.6716	0.6697	1.0000	0.1393	0.2446	0.9682	1.0000	0.2206	0.3614	0.9995
	Family	1.0000	0.8667	0.9286	0.5844	1.0000	0.2167	0.3562	0.9995	1.0000	0.3362	0.5032	0.9996
D4	Subspecies	1.0000	0.4615	0.6316	0.5287	NA	NA	NA	NA	0.9722	0.0461	0.0879	0.9902
	Species	1.0000	0.3722	0.5425	0.5698	0.9697	0.0515	0.0979	0.9823	0.9798	0.0450	0.0861	0.9835
	Genus	1.0000	0.7757	0.8737	0.5672	0.9880	0.1026	0.1859	0.9918	0.9880	0.0976	0.1777	0.9926
	Family	1.0000	0.9041	0.9496	0.5936	1.0000	0.2245	0.3667	0.9932	1.0000	0.2178	0.3577	0.9938
D5	Subspecies	1.0000	0.3846	0.5556	0.9661	NA	NA	NA	NA	1.0000	0.0881	0.1619	0.9251
	Species	1.0000	0.3684	0.5385	0.9761	0.9796	0.1330	0.2341	0.9813	0.9796	0.0777	0.1439	0.9810
	Genus	1.0000	0.5844	0.7377	0.9754	1.0000	0.2356	0.3814	0.9993	1.0000	0.1613	0.2778	0.9995
	Family	1.0000	0.8298	0.9070	0.9728	1.0000	0.3023	0.4643	0.9994	1.0000	0.2583	0.4105	0.9996
D6	Subspecies	1.0000	0.4500	0.6207	0.9948	NA	NA	NA	NA	0.9722	0.0507	0.0963	0.9966
	Species	1.0000	0.3822	0.5531	0.9624	0.9697	0.0659	0.1235	0.9807	0.9798	0.0469	0.0896	0.9809
	Genus	1.0000	0.6975	0.8218	0.9551	0.9880	0.1252	0.2222	0.9967	0.9880	0.1006	0.1826	0.9968
	Family	1.0000	0.9167	0.9565	0.9751	1.0000	0.2472	0.3964	0.9854	1.0000	0.2222	0.3636	0.9855

Table 3: Performance Evaluations on *mock* Datasets.

		HMSC			CLARK			Kraken		
		Recall	Precision	F1 score	Recall	Precision	F1 sore	Recall	Precision	F1 score
D7	Species	0.8182	0.2250	0.3529	0.7273	0.0069	0.0138	0.7273	0.0040	0.0080
	Genus	1.0000	0.6250	0.7692	1.0000	0.0181	0.0357	1.0000	0.0129	0.0254
	Family	1.0000	0.7143	0.8333	1.0000	0.0394	0.0758	1.0000	0.0322	0.0623
D8	Species	0.6667	0.1678	0.2682	0.6944	0.0126	0.0248	0.7222	0.0100	0.0198
	Genus	0.9130	0.5250	0.6667	0.9565	0.0288	0.0558	0.9565	0.0238	0.0464
	Family	0.9091	0.8000	0.8511	1.0000	0.0806	0.1492	1.0000	0.0690	0.1290

the other state-of-the-art methods in terms of accuracy, memory, and runtime.

ACKNOWLEDGMENTS

This work has been supported in part by the following NSF grants: 1447711, 1743418, and 1843025.

REFERENCES

[1] Sasha K Ames, David A Hysom, Shea N Gardner, G Scott Lloyd, Maya B Gokhale, and Jonathan E Allen. 2013. Scalable metagenomic taxonomy classification using a reference genome database. *Bioinformatics* 29, 18 (Sept. 2013), 2253–2260.

[2] FP Breitwieser, DN Baker, and SL Salzberg. 2018. KrakenUniq: confident and fast metagenomics classification using unique k-mer counts. *Genome biology* 19, 1 (2018), 198.

[3] Christiam Camacho, George Coulouris, Vahram Avagyan, Ning Ma, Jason Papadopoulos, Kevin Bealer, and Thomas L Madden. 2009. BLAST : architecture and applications. *BMC Bioinformatics* 10, 1 (2009), 421.

[4] André Corvelo, Wayne E Clarke, Nicolas Robine, and Michael C Zody. 2018. taxMaps: comprehensive and highly accurate taxonomic classification of short-read data in reasonable time. *Genome research* (2018), gr–225276.

[5] Patricia I Diaz, AK Dupuy, L Abusleme, B Reese, C Obergfell, L Choquette, Anna Dongari-Bagtzoglou, Douglas E Peterson, E Terzi, and LD Strausbaugh. 2012. Using high throughput sequencing to explore the biodiversity in oral bacterial communities. *Molecular oral microbiology* 27, 3 (2012), 182–201.

[6] Antonino Fiannaca, Laura La Paglia, Massimo La Rosa, Giovanni Renda, Riccardo Rizzo, Salvatore Gaglio, Alfonso Urso, et al. 2018. Deep learning models for bacteria taxonomic classification of metagenomic data. *BMC bioinformatics* 19, 7 (2018), 198.

[7] Hadrien Gourlé, Oskar Karlsson-Lindsjö, Juliette Hayer, and Erik Bongcam-Rudloff. 2018. Simulating Illumina metagenomic data with InSilicoSeq. *Bioinformatics* 35, 3 (2018), 521–522.

[8] D H Huson, A F Auch, J Qi, and S C Schuster. 2007. MEGAN analysis of metagenomic data. *Genome Res.* 17, 3 (2007), 377–386.

[9] David Koslicki and Daniel Falush. 2016. MetaPalette: a k-mer Painting Approach for Metagenomic Taxonomic Profiling and Quantification of Novel Strain Variation. *mSystems* 1, 3 (May 2016).

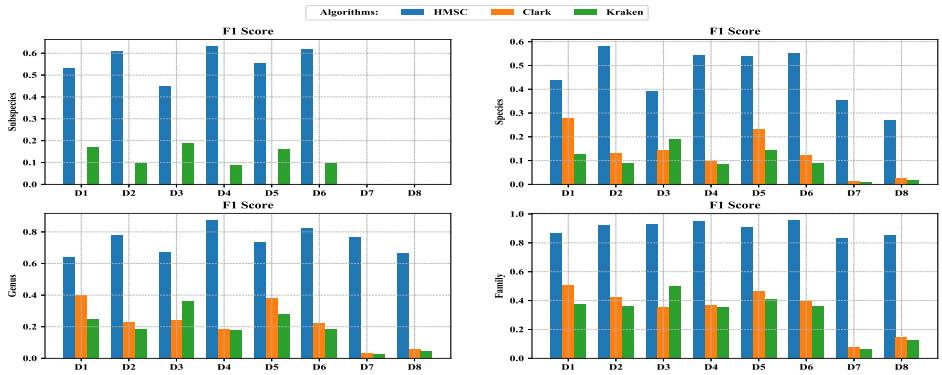
[10] David Koslicki, Simon Foucart, and Gail Rosen. 2014. WGSQuikr: fast whole-genome shotgun metagenomic classification. *PLoS One* 9, 3 (March 2014), e91784.

[11] Bo Liu, Theodore Gibbons, Mohammad Ghodsi, and Mihai Pop. 2010. MetaPhyler: Taxonomic profiling for metagenomic sequences. In *2010 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*.

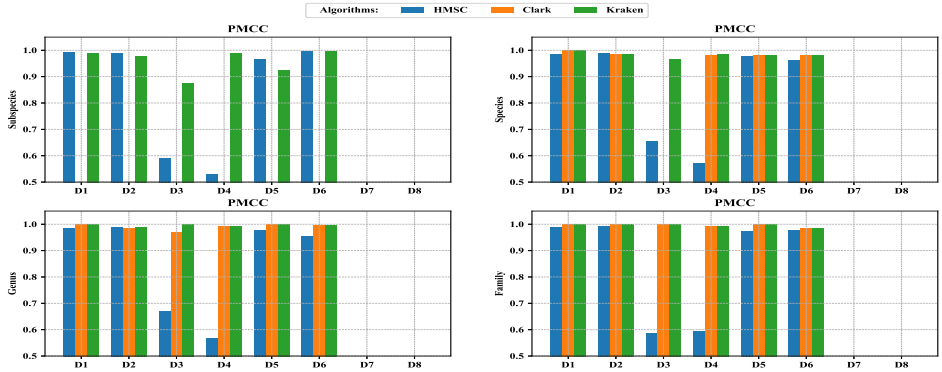
[12] Xinan Liu, Ye Yu, Jinpeng Liu, Corrine F Elliott, Chen Qian, and Jinze Liu. 2017. A novel data structure to support ultra-fast taxonomic classification of metagenomic sequences with k-mer signatures. *Bioinformatics* 34, 1 (2017), 171–178.

[13] Jennifer Lu, Florian P Breitwieser, Peter Thielen, and Steven L Salzberg. 2016. Bracken: Estimating species abundance in metagenomics data.

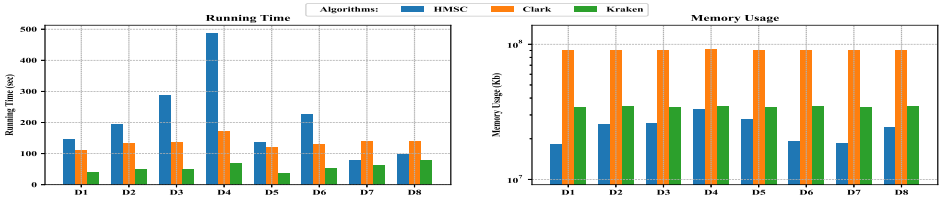
[14] Peter Menzel, Kim Lee Ng, and Anders Krogh. 2015. Kaiju: Fast and sensitive taxonomic classification for metagenomics.



(a) *F1 score* of different algorithms.



(b) *Pearson's coefficient* of different algorithms.



(c) Execution time and memory consumption.

Figure 1: Performance Evaluations.

[15] Aleksandr Morgulis, George Coulouris, Yan Raytselis, Thomas L Madden, Richa Agarwala, and Alejandro A Schäffer. 2008. Database indexing for production MegaBLAST searches. *Bioinformatics* 24, 16 (Aug. 2008), 1757–1764.

[16] André Müller, Christian Hundt, Andreas Hildebrandt, Thomas Hankeln, and Bertil Schmidt. 2017. MetaCache: Context-aware classification of metagenomic reads using minhashing. *Bioinformatics* (Aug. 2017).

[17] Rachid Ounit and Stefano Lonardi. 2016. Higher classification sensitivity of short metagenomic reads with CLARK-S. *Bioinformatics* 32, 24 (Dec. 2016), 3823–3825.

[18] Rachid Ounit, Steve Wanamaker, Timothy J Close, and Stefano Lonardi. 2015. CLARK: fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers. *BMC Genomics* 16 (March 2015), 236.

[19] Jane Peterson, Susan Garges, Maria Giovanni, Pamela McInnes, Lu Wang, Jeffery A Schloss, Vivien Bonazzi, Jean E McEwen, Kris A Wetterstrand, Carolyn Deal, et al. 2009. The NIH human microbiome project. *Genome research* 19, 12 (2009), 2317–2323.

[20] L Schaeffer, H Pimentel, N Bray, P Melsted, and L Pachter. 2017. Pseudoalignment for metagenomic read assignment. *Bioinformatics* 33, 14 (July 2017), 2082–2088.

[21] Shinichi Sunagawa, Daniel R Mende, Georg Zeller, Fernando Izquierdo-Carrasco, Simon A Berger, Jens Roat Kultima, Luis Pedro Coelho, Manimozhiyan Arumugam, Julien Tap, Henrik Bjørn Nielsen, Simon Rasmussen, Søren Brunak, Oluf Pedersen, Francisco Guarner, Willem M de Vos, Jun Wang, Junhua Li, Joël Doré, S Dusko Ehrlich, Alexandros Stamatakis, and Peer Bork. 2013. Metagenomic species profiling using universal phylogenetic marker genes. *Nat. Methods* 10, 12 (Dec. 2013), 1196–1199.

[22] Simon H Tausch, Benjamin Strauch, Andreas Andrusch, Tobias P Loka, Martin S Lindner, Andreas Nitsche, and Bernhard Y Renard. 2018. LiveKraken–Real-time metagenomic classification of Illumina data. *Bioinformatics* 1 (2018), 3.

[23] Duy Tin Truong, Eric A Franzosa, Timothy L Tickle, Matthias Scholz, George Weingart, Edoardo Pasolli, Adrian Tett, Curtis Huttenhower, and Nicola Segata. 2015. MetaPhlAn2 for enhanced metagenomic taxonomic profiling. *Nat. Methods* 12, 10 (Oct. 2015), 902–903.

[24] Derrick E Wood and Steven L Salzberg. 2014. Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biol.* 15, 3 (March 2014), R46.