K2Mem: Discovering Discriminative K-mers from Sequencing Data for Metagenomic Reads Classification

Davide Storato, Matteo Comin

Abstract—The major problem when analyzing a metagenomic sample is to taxonomically annotate its reads to identify the species they contain. Most of the methods currently available focus on the classification of reads using a set of reference genomes and their k-mers. While in terms of precision these methods have reached percentages of correctness close to perfection, in terms of recall (the actual number of classified reads) the performances fall at around 50%. One of the reasons is the fact that the sequences in a sample can be very different from the corresponding reference genome, e.g. viral genomes are highly mutated. To address this issue, in this paper we study the problem of metagenomic reads classification by improving the reference k-mers library with novel discriminative k-mers from the input sequencing reads. We evaluated the performance in different conditions against several other tools and the results showed an improved F-measure, especially when close reference genomes are not available.
Availability: https://github.com/CominLab/K2Mem

Index Terms—Metagenomic Reads Classification, Discriminative k-mers, Minimizers.

1 Introduction

Metagenomics is the study of the heterogeneous microbes samples (e.g. soil, water, human microbiome) directly extracted from the natural environment with the primary goal of determining the taxonomical identity of the microorganisms residing in the samples. It is an evolutionary revise, shifting focuses from the individual microbe study to a complex microbial community. The classical genomic-based approaches require the prior clone and culturing for further investigation [1]. However, not all Bacteria can be cultured. The advent of metagenomics succeeded to bypass this difficulty. Microbial communities can be analyzed and compared through the detection and quantification of the species they contain [2]. Then, the inner structure of a microbial community can be unveiled using ecological measures such as species diversity, richness and uniformity. In this paper, we will focus on the detection of species in a sample using a set of reference genomes, e.g. Bacteria and Virus. The reference-based metagenomics classification methods can be broadly divided into three categories: (1) alignment-based methods, (2) marker-based methods, where certain specific marker sequences are used to identify the species. (3) sequence-composition-based methods, which are based on the nucleotide composition (e.g. k-mers usage). Traditionally, the first strategy was to use BLAST [3] to align each read with all sequences in GenBank. Later, faster methods have been deployed for this task, popular examples are MegaBlast [4] and Megan [5]. However, as the reference databases and the size of sequencing data sets have grown, alignment has become computationally infeasible, leading to the development of metagenomics classifiers that provide much faster results.

Marker-based methods use clade specific marker genes as a taxonomic reference, so that the identification of one of these genes can be used as evidence that a given taxa is present. This allows faster assignment because the database of marker genes is far smaller than a database of the full genomes for all species. Popular examples of marker gene methods are MetaPhlAn [6], Phylosift [7].

The fastest and most promising approaches belong to the composition-based one [8]. The basic principles can be summarized as follows: each genome of reference organisms is represented by its k-mers, and the associated taxonomic label of the organisms, then the reads are searched and classified throughout this k-mers database. For example, Kraken [9] constructs a data structure that is an augmented taxonomic tree in which a list of significant k-mers are associated to each node, leaves and internal nodes. Given a node on this taxonomic tree, its list of k-mers is considered representative for the taxonomic label and it will be used for the classification of metagenomic reads. CLARK [10] uses a similar approach, building databases of species- or genus-level specific k-mers, and discarding any k-mers mapping to higher levels. The precision of these methods is as good as MegaBlast [4], nevertheless, the processing speed is much faster [8]. Several other composition-based methods have been proposed over the years. In [11] the number of unassigned reads is decreased through reads overlap detection and species imputation. Centrifuge and Kraken 2 [12], [13] try to reduce the size of the k-mer database with the use respectively of FM-index and minimizers. The sensitivity can be improved by filtering uninformative k-mers [14], [15] or by using spaced seeds instead of k-mers [16].

The major problem with these reference-based metage-

• *D. Storato is with the Department of Molecular Medicine, University of Padua, Padua, 35100, Italy*
E-mail: davide.storato@unipd.it
• *M. Comin is with the Department of Information Engineering, University of Padua, Padua, 35100, Italy*
E-mail: matteo.comin@unipd.it

nomics classifiers is the fact that most Bacteria found in environmental samples are unknown and cannot be cultured and separated in the laboratory [17]. As a consequence, the genomes of most microbes in a metegenomic sample are taxonomically distant from those present in existing reference databases. This fact is even more important in the case of viral genomes, where the mutation and recombination rate is very high and as a consequence, the viral reference genomes are usually very different from the other viral genomes of the same species.

For these reasons, most of the reference-based metagenomics classification methods do not perform well when the sample under examination contains strains that are different from the genomes used as references. Indeed, e.g. CLARK [10] and Kraken [9] report precisions above 95% on many datasets. On the other hand, in terms of recall, i.e. the percentage of reads classified, both Clark and Kraken usually show performances between 50% and 60%, and sometimes on real metagenomes, just 20% of reads can be assigned to some taxa. In this paper we address this problem and we propose a metagenomics classification tool, named *K2Mem*[1], that is based, not only on a set of reference genomes but also it uses discriminative k-mers from the input metagenomics reads to improve the classification. The basic idea is to add memory to a classification pipeline, so that previously analyzed reads can be of help for the classification.

2 Methods

To improve the metagenomics classification our idea is based on the following considerations. All reference-based metagenomics classification methods need to index a set of reference genomes. The construction of this reference database is based on a set of genomes, represented by its k-mers (a piece of the genome with length k), and the taxonomic tree. For example, Kraken [9] constructs a data structure that is an augmented taxonomic tree, in which a list of discriminative k-mers is associated with each node, leaves and internal nodes. Given a node on this taxonomic tree, its list of k-mers is considered representative for the taxonomic label and it is used for the classification of metagenomic reads. However, for a given genome only a few of its k-mers will be considered discriminative. As a consequence, only the reads that contains these discriminative k-mers can be classified to this species.

Given a read with length n, each of its $n-k+1$ k-mers have the first $k-1$ bases in common with the previous k-mer, except the first k-mer. Furthermore, it is possible that reads belonging to the input sequencing data can have many k-mers in common.

As can be seen in Fig. 1, in this example we have three reads containing the same k-mer (in red) but only one is classified thanks to the presence in the read of a discriminative k-mer (in green), with a taxonomy ID associated, contained in the classifier's database. The second read could not be classified because none of its k-mers are in the k-mer reference library, as there is a mutation (in bold) with respect to the reference genome. However, the k-mers of the first read, that are not present in the classifier's database, can belong to the same species to which the read classified belongs to. With reference to Fig. 1, if we associate to the shared k-mer (the red one) the taxonomy ID of the first read then, we can classify the other two reads. Thus, using the above considerations, one can try to extend the taxonomy ID of a classified read to all its k-mers.

The idea is to equip the classifier with memory from previous classifications, thus adding novel discriminative k-mers found in the input sequencing data. To obtain this memory effect, one needs to modify a given classifier with additional data structures and a new classification pipeline. Note that, this idea can be applied to any reference-based metagenomics classifiers that are based on a database of k-mers. In this paper we choose to use Kraken 2 [13], that was recently introduced, and that is reported to be the current state of the art. Before to describe our classification tool, for completeness here we give a brief introduction of Kraken 2 to better understand our contribution.

2.1 Background on Kraken 2

Kraken 2 is an improved version of the classifier Kraken [9] regarding memory usage and classification time. These improvements were obtained thanks to the use of the minimizers and a probabilistic compact data structure, instead of the k-mers and a sorted list used in Kraken. Given a string s of length k, and a value l, with $l < k$. The minimizer (k,l) of the string s is the smallest l-mer, according to the alphabetical order, that appears in s. The minimizers method is very versatile and is used, instead of k-mers, in various ways in many bioinformatics programs to reduce the total computational cost or the memory usage, see [19] for a review.

Instead of utilizing the complete genome as reference, Kraken 2 considers only its minimizers $(k=35, l=31)$, thus a genome sequence is alternatively represented by a set of l-mers, which plays a role of efficiently indexing a large volume of target-genomes database, e.g., all the genomes in RefSeq. This idea is borrowed from alignment-free methods [20] and some researchers have verified its availability in different applications. For instance, the construction of phylogenetic trees, traditionally is performed based on multiple-sequence alignment, whereas with alignment-free methods it can be carried out on the whole genomes [21], [22]. Recently some variations of k-mers-based methods have been devised for the detection of enhancers in ChIP-Seq data [23], [24], [25], [26], [27], entropic profiles [28], [29], and NGS data compression [30], [31], [32]. Also, the assembly-free comparison of genomes and metagenomes based on NGS reads and k-mers counts has been investigated in [33], [34], [35], [36]. For a comprehensive review of alignment-free measures and applications we refer the reader to [20].

At first, Kraken 2 needs to build a database starting from a set of reference genomes. To build the database Kraken 2 downloads from the NCBI the taxonomy and reference sequences libraries required. With the taxonomy data, Kraken 2 builds a tree where each node is associated to a taxonomy ID. In each tree node, a list of minimizers is stored that is useful for the classification. Precisely, for each minimizer (k=35, l=31) if it is unique to a reference sequence

1. A preliminary version of this work appeared in the proceedings of ISBRA 2020 [18].

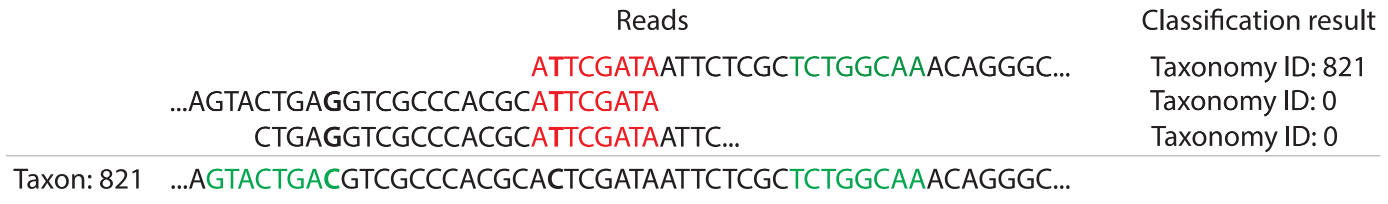
Reads
Classification result
ATTCGATAATTCTCGCTCTGGCAAACAGGGC...
Taxonomy ID: 821
...AGTACTGAGGTCGCCCACGCATTCGATA
Taxonomy ID: 0
CTGAGGTCGCCCACGCATTCGATAATTC...
Taxonomy ID: 0
Taxon: 821
...AGTACTGACGTCGCCCACGCACTCGATAATTCTCGCTCTGGCAAACAGGGC...

Fig. 1. Example of a reference-based metagenomics classifier behaviour. In red the k-mer in common between the reads, in green the k-mer associated to a species' taxonomy ID (in this case 821) present in the classifier's database, and in bold the mutations' positions in the reads. A taxonomy ID of zero indicates that the classifier wasn't able to classify the read.

then, it is associated with the node with the sequence's taxonomy ID. Instead, if the minimizer belongs to more than one reference sequence then, the minimizer is moved to the node with taxonomy ID equals to the Lowest Common Ancestor (LCA) of the two sequences it belongs. All the minimizer-taxonomy ID pairs are saved in a probabilistic Compact Hash Table (CHT) which allows a reduction of memory with respect to Kraken. Kraken 2 uses also a spaced seed mask to the minimizer and calculates a compact hash code, which is then used as a search query in its Compact Hash Table, see [13] for more details.

Once the database is built, Kraken 2 classifies the input reads in a more efficient manner than Kraken due to the smaller number of accesses to the CHT map. This is due to the fact that only distinct minimizers from the read trigger the research in the map. When the minimizer is queried in the compact table, the node ID of the augmented tree is returned, and the counter in the node is increased. Once all the minimizers have been analyzed, the read is classified by choosing the deepest node from the root with the highest-weight path in the taxonomy tree. The use of spaced seeds and minimizers allow Kraken 2 to improve the sensitivity.

2.2 K2Mem

Here we present K2Mem (Kraken 2 with Memory) a classifier based on Kraken 2. In order to implement the idea explained above, K2Mem needs to detect new discriminative minimizers, to store them in memory and to use this additional information in the classification of reads. The new classification pipeline of K2Mem discovers novel discriminative minimizers from the input sequencing data and it saves them in a map of additional minimizers.

The data structure used to store these additional minimizers is an unordered map that stores pairs composed of the novel discriminative minimizer, not present in the classifier's database, and the taxonomy ID associated to the read that contains the minimizer. An unordered map is an associative container that contains key-value pairs with a unique key. The choice of this structure is due to the fact that search, insertion and removal of elements have average constant-time complexity. Internally, the elements are not ordered in any particular order but are organized in buckets. Which bucket an element is placed into depends entirely on the hash of its key. This allows fast access to the single element, once the hash is computed, it refers to the exact bucket where the element has been inserted. The key and value are both 64-bit unsigned integer. This choice was made to keep the complete minimizers (l=31) on the map without loss of information due to the CHT hash function and to contains the taxonomy ID in case the number of taxonomy tree nodes increases in future.

K2Mem has two main steps, in the first phase all reads are processed and novel discriminative minimizers are discovered and stored in the additional minimizers map. In the second phase, the same input reads are re-classified using the Compact Hash Table and also the additional minimizers obtained in the first phase.

An overview of the first phase, the discovery of novel discriminative minimizers, is reported in Fig. 2. The population of the additional minimizers map works as follow: for each read, its minimizers (k=35, l=31) are computed one at a time and, for each of them, the Compact Hash Table (CHT) is queried (1). If it returns a taxonomy value equal to zero, then the additional map is queried if it is not empty (2). If the minimizer is not found in the additional minimizers map, this means that the minimizer is not in the Kraken 2's database and no taxonomy ID has been assigned to it or is the first time the minimizer is found. In that case, the minimizer is added to a temporary list of not taxonomically assigned minimizers (3). Instead, if the CHT or the additional minimizers map query returns a taxonomy ID not equal to zero, then the taxonomy ID count is updated (4). Then, the read is classified (5), based on the highest-weight path in the taxonomy tree, and the resulting taxonomy ID is checked if it is at the species level or below (6). If it is, then the minimizers in the unknown minimizers list are added to the additional minimizers map with key the minimizer and value the taxonomy ID obtained by the read classification. If the minimizer is already in the additional map the LCA of the input and stored taxonomy IDs value is saved. Instead, if the taxonomy ID obtained after the read classification is at a level above the species then, the minimizers are not added and the list is emptied. In the first phase, this procedure is repeated for all the reads in the dataset.

Following the example in Figure 1, let's assume that we are processing the first read, and that the first k-mer is a minimizer. The first minimizer, in red, is not present in Kraken 2's database, and the additional minimizers map is initially empty. Thus, this minimizer is detected in phase 3 and added to the unknown minimizers list. Next, in phase 5, the read is classified at species level, with taxonomy ID of 821. Finally, in phase 6 the newly discovered minimizer and its taxonomy ID are added to the additional minimizers map. The pseudocode for the discovery of novel discriminative minimizers is reported in Algorithm 2.1.

Then, once the population of the additional minimizers map is completed, in the second phase all reads are re-

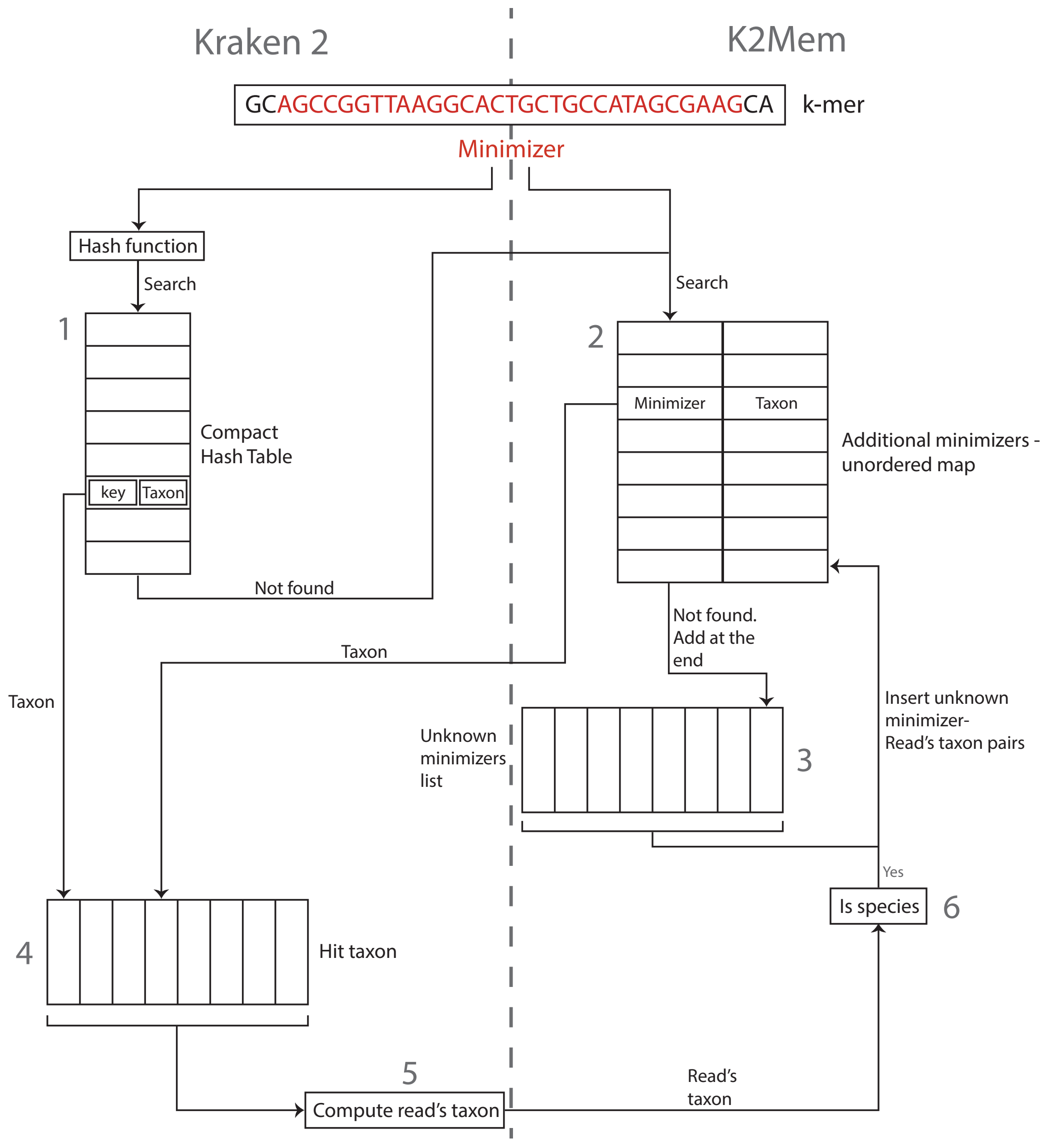

Fig. 2. An overview of K2Mem and the discovery of additional minimizers.

```
foreach read $R \in input\_file$ do
    foreach minimizer $m \in R$ do
        taxon ← CHT(m);
        if $taxon = 0$ and AM not empty then
            // Additional Map (AM)
            taxon ← AM(m);
        if $taxon = 0$ then  // Unknown Minimizer List (UML)
            UML ← m;
    read_taxon ← compute_taxon;
    if read_taxon is species then
        foreach $m \in UML$ do
            AM ← <m, read_taxon>;
```

Algorithm 2.1: Discovery of additional minimizers phase pseudocode.

```
foreach read $R \in input\_file$ do
    foreach minimizer $m \in R$ do
        taxon ← CHT(m);
        if $taxon = 0$ and AM not empty then
            // Additional Map (AM)
            taxon ← AM(m);
    read_taxon ← compute_taxon;
```

Algorithm 2.2: Classification phase pseudocode.

classified. In this phase the classification only uses the CHT and the additional minimizers map to classify the reads in the same dataset, but without to update the additional map content. With these additional minimizers, the dataset is processed to obtain a better classification. K2Mem starts querying the CHT for each minimizer in the read (1). If the minimizer is not found, the additional map is queried (2). If the minimizer is not present in the latest map, a taxon of zero is assigned to the minimizer. Instead, if in the CHT or additional map the minimizer is found, the associated taxon is returned and the taxon's number of hit is updated (4). Once all the read's minimizers are analyzed the read's taxon is computed (5), classifying the read. When K2Mem ends, it generates the same output files as Kraken 2. The pseudocode of the classification phase is reported in Algorithm 2.2.

3 Results

To analyze the performance of K2Mem we compare it with five of the most popular classifiers: Centrifuge [12], CLARK [10], Clior [11], Kraken [9], Kraken 2 [13], and KrakenUniq [37]. We test all these tools on several simulated datasets with different Bacteria and Viral genomes from NCBI and on two real datasets. The experimental setup is described in the next section.

3.1 Experimental setup

In this section we briefly describe the strain exclusion experiment, the datasets, and the evaluation measures used to compare the performance of K2Mem with the other classifiers. To assess the performance of K2Mem we follow the experimental setup of Kraken 2 [13], that is the strain exclusion experiment.

3.1.1 Strain exclusion

The strain exclusion experiment's data were generated as done by the Kraken 2's author in [13]. Specifically, the generation of these data occurs in the way explained below.

It stars by downloading the reference genomes and the taxonomy from the NCBI's database. The reference genomes are generated from the Archaeal, Bacteria, and Viral genomes. From the NCBI's taxonomy, the files containing the taxonomic tree and the associations of GenBank identifier to taxonomy ID are downloaded. From these genomes a subset is selected containing 40 Bacteria genomes and 10 Viral genomes. These elements will be the origin strains for the strain exclusion experiment, used to create the simulated datasets. At this point a reference genomes set is created taking all the data downloaded from the NCBI and removing the origin strains previously chosen. Once the reference genomes set is created, Mason 2 [38] is used for simulate 100 bps paired-end Illumina sequence data from the origin strains. Specifically, the Mason 2's mason_simulator command is used with Illumina error profile for the simulation of the sequence's errors. That means that the generated sequences contain an error rate of 0.4% mismatches, 0.005% insertions and 0.005% deletions. The reads obtained from the origin strains are concatenated in a single file and the truth file of the simulated reads is generated for each library. This setup tries to mimic a realistic scenario in which a reference genome is available, but the sequenced genome is a different strain from the same species.

3.1.2 Datasets

Different simulated and real metagenomics datasets are used to compare and evaluate K2Mem's performance. Ten simulated datasets were created using the origin strains obtained from the strain exclusion experiment explained above. Of these datasets, seven are built by varying the number of reads from 50k to 100M. These datasets are used to test the impact of sequencing coverage on the performance of the tools under examination. We also constructed other three datasets, all with the same number of reads 100M, but with different mutation rates from the original strains: 2%, 5%, and 10%. With these datasets, we evaluate another scenario in which a close reference genome is not available.

In addition to the simulated datasets, two real datasets are used to test K2Mem performance. These datasets are the SRR1804065 and the marine raw long reads, downloaded from the NCBI and CAMI2 [39] website respectively. The real metagenome SRR1804065 is a DNA tool sample from the Human Microbiome Project (HMP), and it has been used for testing in many studies [14], [40], [41]. Since the "ground truth" is not available for a real metagenome we used the same evaluation procedure as in [40] and other studies. Solely with the purpose to evaluate our method on real data, we use BLAST to map the reads against all reference genomes, and filter out the reads that maps on two or more genomes. After this filter we keep the reads that can be mapped uniquely to one genome, with a high confident match of at least 95% identity. These reads will be used as "ground truth" for testing. Since this filtering step can alter the abundance ratios of species, on real datasets we do not evaluate the PCC of the abundance profile. The resulting dataset SRR1804065 comprises 5.5M reads of length 100 and 2537 species, whereas the marine dataset contains 1.1M reads of length between 1000 and 3000 bases and 4956 species. In Table 1 is presented a summary of the simulated and real datasets used for testing.

Dataset	No. of reads	reads length	No. of species
Simulated (x10)	[50k-100M]	100	40 Bacteria 10 Virus
CAMI2 Marine	1150123	[1000-3000]	4956
SRR1804065	5500983	100	2537

TABLE 1
A summary of the datasets used for testing: 10 simulated dataset (7 varying the number of reads and 3 varying the mutation rate); two real datasets.

3.1.3 Evaluation measures

To compare K2Mem with the other tools we use the same evaluation metrics as in [13], [14]; precisely we use Sensitivity, Positive Predicted Value (PPV), F-measure, and Pearson Correlation Coefficient (PCC) of the abundance profile. To compute these measures we use the same methodology used in [13] as following explained. We firstly compare the classifier's results with the truth file of the dataset classified, at a specific taxonomy level, to obtain the number of reads belonging to the following categories: true positive (TP), false negative (FN), and false positive (FP). A read belongs to the TP category if its taxonomy classification rank is the same or is a descendant of the truth rank. A read belongs to the FN category if the classifier fails to classify the sequence or if its taxonomy classification rank is an ancestor of the truth rank. Lastly, a read belongs to FP if its classification is incorrect; that is, it is not in the true taxonomy of origin, it is not an ancestor or a descendant of the truth rank. Using these categories we define the sensitivity as the proportion of the number of reads correctly classified over the total number of reads. We define the PPV as the proportion of the number of reads correctly classified among the number of classified reads. The F-measure is defined as the harmonic mean of sensitivity and PPV. The Pearson Correlation Coefficient is defined in the following manner: Given paired data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ consisting of n pairs, PCC is defined as:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \overline{x}) \cdot (y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2} \cdot \sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}}. \qquad (1)$$

Where: n is the number of species, x_i, y_i are the number of species i correctly classified and present in ground truth respectively, and $\overline{x}$ and $\overline{y}$ is the sample mean of x and y respectively. The sample mean of x (analogously of y) is computed as: $\overline{x} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$. The Pearson Correlation Coefficient reflects the ability of a classifier to detect the correct abundance rate of species.

3.2 Performance Evaluation

In this section, we analyze the performance results at the genus and species levels of K2Mem with respect to the other classifiers. All tools are used with the default parameters and run in multithreading using 16 threads. Their databases are built using the same set of reference genomes obtained from the strain exclusion experiment.
The obtained results are reported below in different figures to better understand the performance and the impact of the different configurations.

3.2.1 Overall Results for Bacteria and Virus

In the first experiments we test the overall performance of all tools on the largest simulated dataset with 100M reads. We analyze the 100M dataset as it is the most relevant case due to its dimension. On this dataset Clior runs out of memory on a machine with 2 TB of RAM.

In Figure 3 and 4 are shown the full results obtained with the 100M reads dataset analyzed at genus level for Bacteria and Virus respectively.

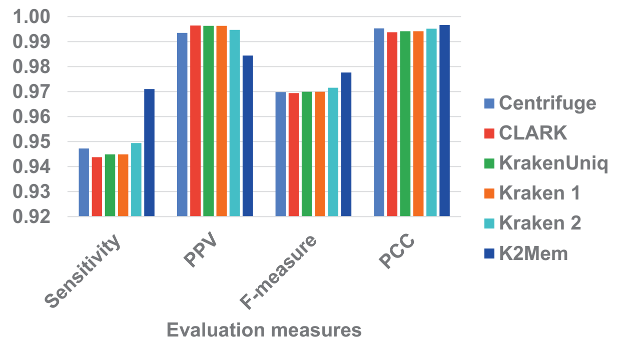

Fig. 3. Bacteria evaluation at genus level on the 100M reads dataset.

As it can be seen in Figure 3, with Bacteria K2Mem obtains an F-measure improvement of at least 0.5 percentage points (pps) respect to the closest competitor, Kraken 2, and the other classifiers. This improvement is due to an increase of sensitivity of at least 2 pps despite a worsening of the PPV of about 1 pps respect to the other tools. Moreover, K2Mem obtains the best PCC value with an improvement of at least 0.1 pps respect to Centrifuge and Kraken 2.

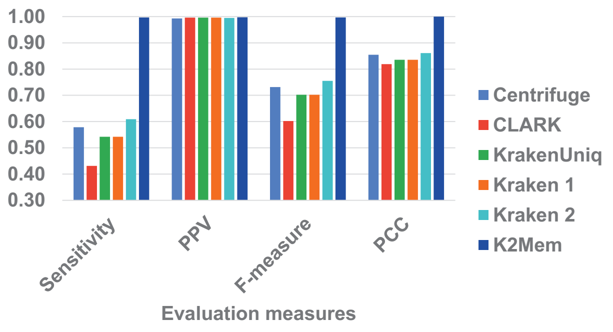

Fig. 4. Viruses evaluation at genus level on the 100M reads dataset.

On the viral dataset, as it can be observed in Figure 4, K2Mem obtains an F-measure improvement of almost 40 pps respect to Kraken 2 and the other tools. This improvement is due to an increase of sensitivity, whereas the PPV of all tools is to close to 1. Moreover, K2Mem shows the best PCC with an improvement of about 13 pps respect to Centrifuge and Kraken 2.

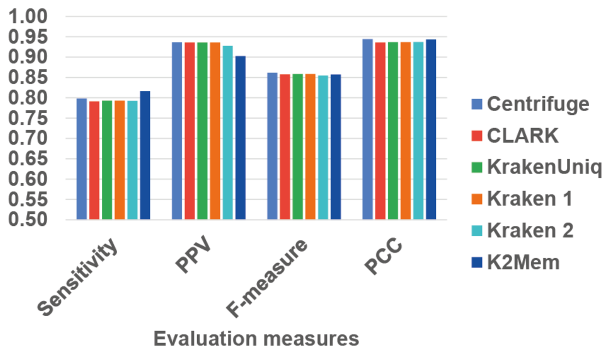

Fig. 5. Bacteria evaluation at species level on the 100M reads dataset.

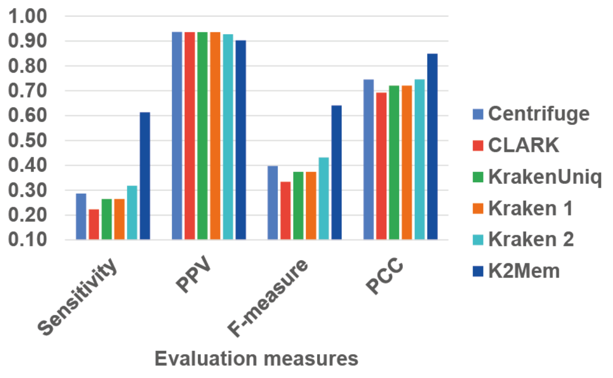

Fig. 6. Viruses evaluation at species level for the 100M reads dataset.

In Figure 5 and Figure 6 are shown the full results obtained with 100M reads dataset analyzed at the species level for Bacteria and Virus, respectively. The classification at species level is clearly more difficult than at genus level, and the results reported are lower for all tools with respect to the previous Figures.

As can be seen in Figure 5, on the Bacteria dataset K2Mem obtains a balanced sensitivity improvement and PPV worsening that gives a slight worsening in F-measure with respect to the best method, precisely of at most 0.4 pps. However, the F-measure value is very close to the other tools, and slightly better than Kraken 2. A Similar consideration applies to the PCC values.

On the viral dataset, as can be seen in Figure 6, K2Mem shows an F-measure improvement of at most 20 pps respect to the other tools. This improvement is due to sensitivity improvement. Furthermore, K2Mem gives the best PCC value with a difference of at least 10 pps respect to the other tools. In summary, with the results reported above, we can observe that thanks to the additional information provided by the new discriminative minimizers K2Mem obtains a moderate improvement in Bacteria's classification and a significant improvement in Virus' classification.

3.2.2 Results varying the number of reads

In these experiments we evaluate the impact of sequencing coverage by varying the number of reads in the dataset from 50k to 100M reads. In the following Figures we report the F-measure as the best means of comparison. In Figure 7 and 8 are shown the F-measure values, evaluated at genus level, obtained varying the number reads in the dataset for Bacteria and Virus respectively. For Bacteria, as it can be seen in Figure 7, K2Mem gets better F-measure values than the other tools as the number of reads increases; obtaining improvements up to almost 1 pps. This improvement is given mainly from the increase of sensitivity.

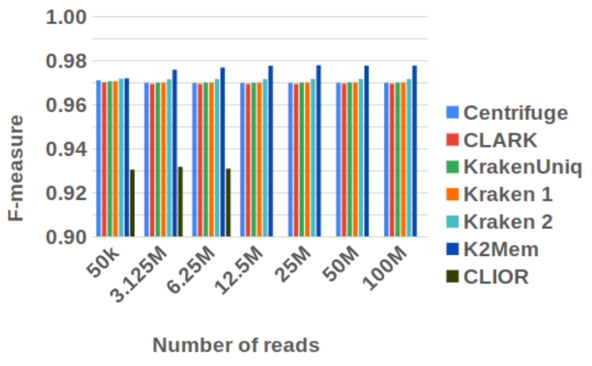

Fig. 7. F-measure values for Bacteria at genus level varying the number of reads.

For Virus, as it can be seen in Figure 8, K2Mem achieves greater F-measure improvement than Bacteria, always thanks to an increase of the sensitivity. It is interesting to note that the performances of all other tools are independent of the size of the dataset.

In Figure 9 and 10 are shown the F-measure values, evaluated at the species level, obtained varying the number of reads in the dataset for Bacteria and Virus respectively.

For Bacteria, as shown in Figure 9, K2Mem does not report an improvement on larger datasets and the F-measurse remains constant, in line with the other tools.

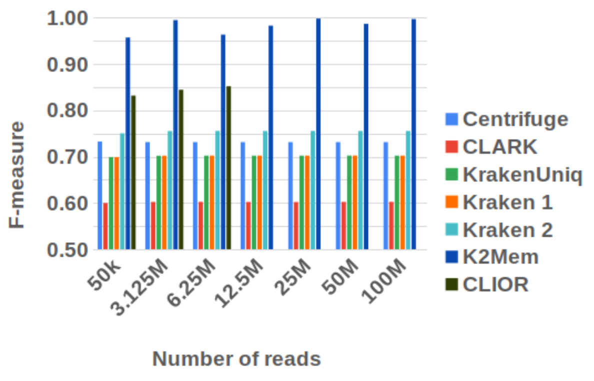

Fig. 8. F-measure values for Virus at genus level varying the number of reads.

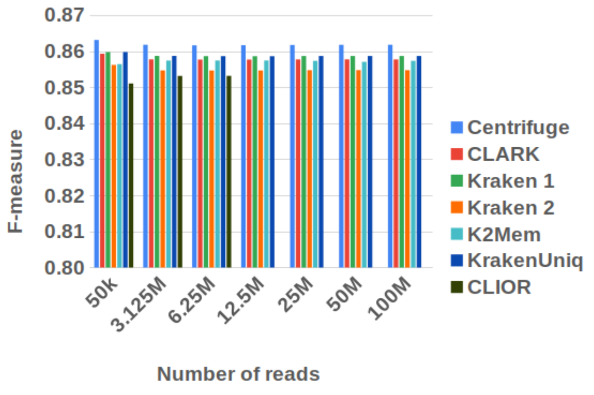

Fig. 9. F-measure values for Bacteria at species level varying the number of reads.

With Virus, as can be seen in Figure 10, K2Mem obtains an improvement of the F-measure value, around 20 pps, respect to Kraken2 and the other tools. This improvement is given mainly from sensitivity increasing. Clior shows similar performance on the Viral datset, Figures 8 and 10. However, the improvement of the F-measure is lower, moreover Clior can not handle large datasets with 12.5M reads or more. In summary, we can observe that for K2Mem the greater the amount of data the better the classification

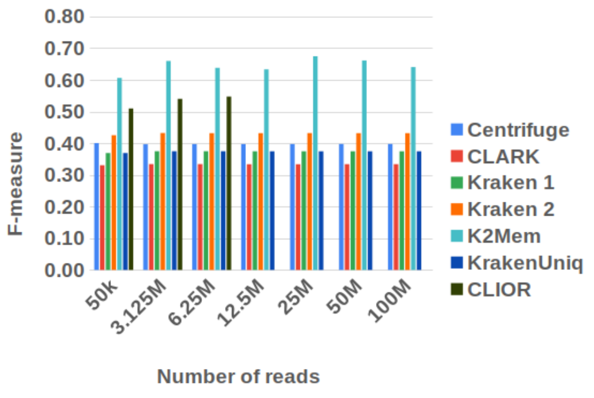

Fig. 10. F-measure values for Virus at species level varying the number of reads.

results, due to a greater possibility of finding new discriminative minimizers. This behaviour is more evident on the Viral data where the number of unclassified reads is higher for all the other tools.

3.2.3 Results for Strains distant from the reference genomes

In this experiments we test the realistic situation in which the genomes sequenced do not have a close relative in the database of reference genomes. In order to evaluate this scenario we use again the strain exclusion setup, but this time the reads are simulated with a different mutation rate with respect to the reference strain genomes.

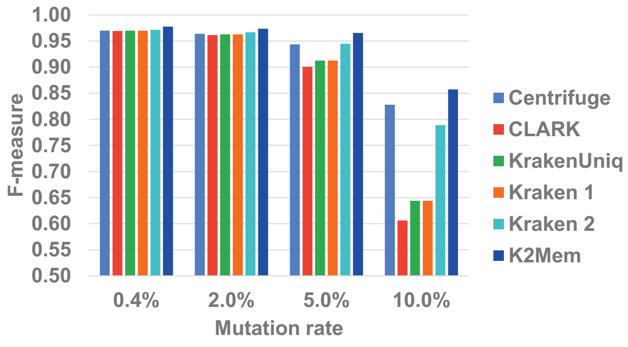

Fig. 11. F-measure values for Bacteria at genus level varying the mutation rate.

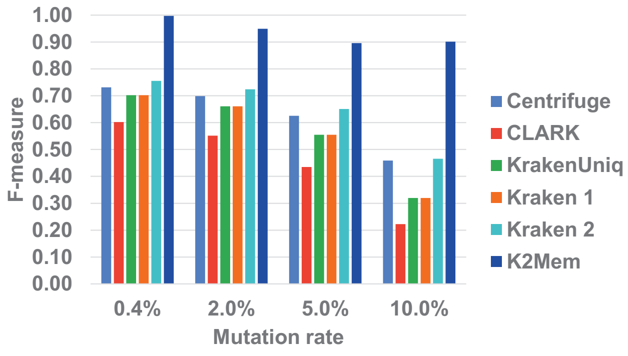

Fig. 12. F-measure values for Virus at genus level varying the mutation rate.

In Figure 11 and 12 are shown the F-measure values obtained for the 100M reads dataset while varying the mutation rate, for Bacteria and Virus respectively. The mutation rate respect to the origin strains rages between 0.4% to 10%.

As it can be seen in Figure 11, the first observation is that the performance of all tools decreases as the mutation rate increases. This is expected because the reference genomes are no longer similar to the genomes in the sample data. However, K2Mem obtains the best F-measure values in all cases. Moreover, the F-measure improvement increases up to 2.5 pps w.r.t. Centrifuge and 7 pps with Kraken 2, as the mutation rate increases.

With Virus, as reported in Figure 12, K2Mem has the same behaviour described for the Bacteria, but with a bigger performance gap up to 45 pps respect to Kraken 2. In Figure 13 and Figure 14 are shown the F-measure values, evaluated at the species level, obtained varying the mutation rate in the Bacteria and Virus dataset, respectively.

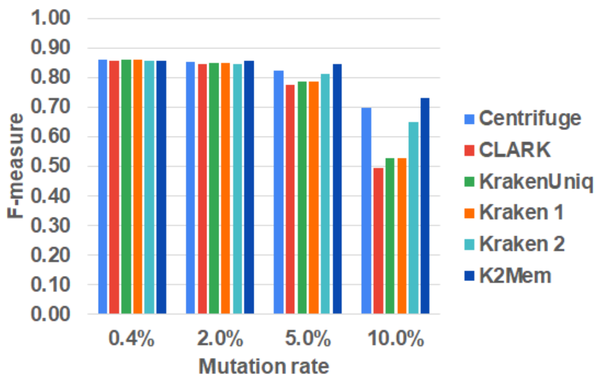

Fig. 13. F-measure values for Bacteria at species level varying the mutation rate.

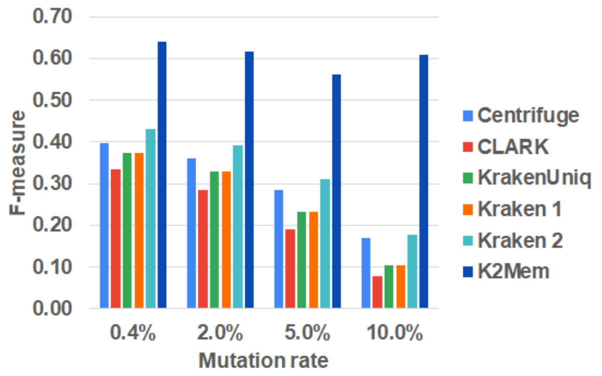

Fig. 14. F-measure values for Virus at species level varying the mutation rate.

For the Bacteria, as can be seen in Figure 13, K2Mem has the same behaviour explained at genus level.

As reported in Figure 14, with Virus at the species level, K2Mem has the same performance described for the Virus evaluated at the genus level. That is, K2Mem obtains best F-measure values w.r.t. the other tools with a difference of at least 20 pps.

In summary, the increase in the mutation rate leads to worse performance for all tools as expected. However, K2Mem is the classifier that has suffered less from the presence of mutations in the sample, and it achieved the best performance over all other methods. Moreover, this behaviour is consistent over all configurations for both species and genus levels and both Bacteria and Virus.

3.2.4 Results on real datasets

In this section we analyze the performance of all tools on two real datasets: SRR1804065 and CAMI2 marine.

In Figure 15 and Figure 16 are shown the results obtained with the CAMI2 marine and SRR1804065 datasets respectively, analyzed at the genus level. Hereafter the PCC values are not evaluated since they can be biased by the ground truth construction.

As it can be seen in Figure 15, with the marine dataset K2Mem is the second best tool with a worsening in F-

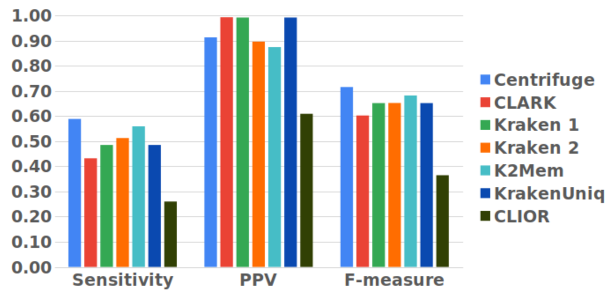

Fig. 15. Evaluation at genus level on the CAMI2 marine datasets.

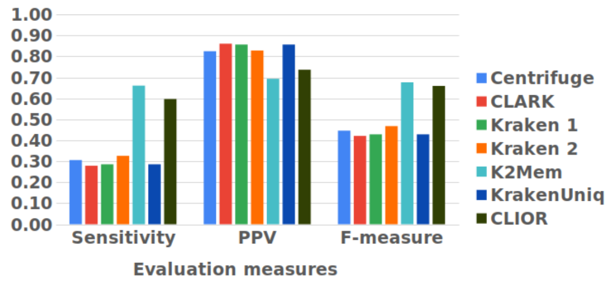

Fig. 16. Evaluation at genus level on the SRR1804065 datasets.

measure of 5 pps with respect to Centrifuge, the first one, and an improvement of at least 2 pps with respect to the remaining classifiers, including Kraken 2. On the marine dataset, with its long reads, we think that the Centrifuge's better performance are due to the fact that it searches k-mers of variable length, as opposed to the other tools based on fixed length k-mers.

With the SRR1804065 dataset, as reported in Figure 16, K2Mem obtains the best F-measure value with an increment of at least 20 pps respect to most of the other tools. The improvement is due to a sensitivity increasing of at least 30 pps and a PPV decreasing of at most 20 pps. On this dataset Clior shows performance similar to K2Mem, only slightly lower. However, the amount of RAM (780 GB) and time required by Clior is extremely high, see Figure 18.

In summary, on these two real datasets K2Mem obtains very good results, and it always improves the performance of Kraken 2.

3.2.5 *Execution time and memory usage*

In this section we evaluate the computational resources required for all tools. The execution time and memory usage of each tool during the classification of the simulated datasets are shown in Figure 17. For this analysis, the execution time and memory usage values are reported for the largest simulated dataset with 100M reads.

In Figure 18 are reported the execution time and memory usage for the real dataset SRR1804065. For the SRR1804065 dataset, K2mem has a behaviour similar to that of the 100M reads dataset, with a low memory footprint and one of the best execution time. The execution time of K2Mem, as expected, it is higher than Kraken 2 due to the new discriminative minimizers search phase. However, the classification time is in line with the other tools. As for the memory usage K2Mem requires slightly more memory

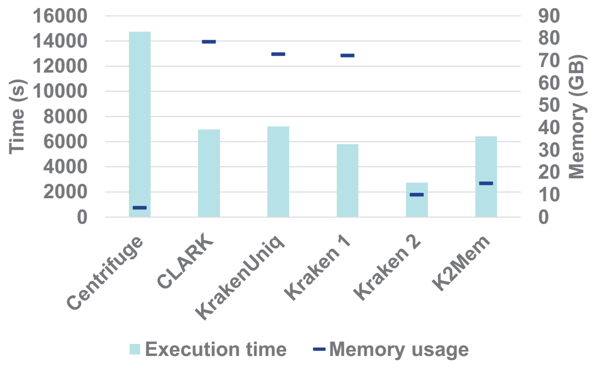

Fig. 17. Execution time and memory usage for the simulated dataset with 100M reads.

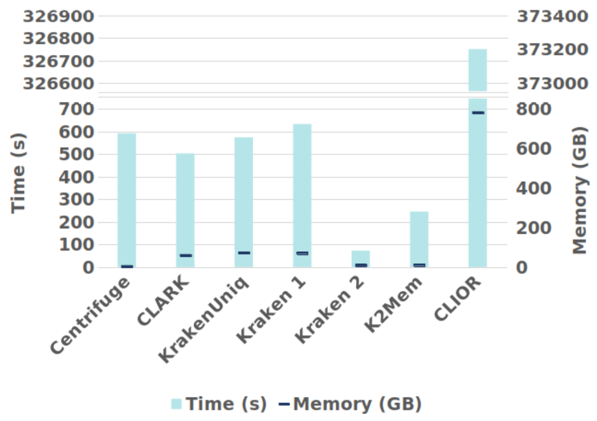

Fig. 18. Execution time and memory usage for the SRR1804065 dataset.

than Kraken 2, due to the new map. For example, on the largest dataset with 100M reads, K2Mem requires only 15 GB of RAM, whereas Kraken 2 9.98 GB. On the real dataset, Figure 18, K2Mem uses only 11.5 GB of RAM, and Kraken 2 9.7 GB. In summary, the computing time and the memory requirements of K2Mem are in line with the best tools.

4 Conclusion

We have presented K2Mem, a classifier based on Kraken 2 with a new classification pipeline and an additional map to store new discriminative k-mers from the input sequencing reads. The experimental results have demonstrated that K2Mem obtains higher values of F-measure, mainly by an improved sensitivity, and PCC respect to the most popular classifiers thanks to the greater number of reference k-mers available during the classification. We showed that the performance improvement of K2Mem increases as the size of the input sequencing data grows, or when reads originate from strains that are genetically distinct from those in the reference database. On real datasets K2Mem obtains very good results, and it always improves the performance of Kraken 2. As possible future developments it could be interesting to increase the PPV, e.g. using unique k-mers, to speed up the classification algorithm with a better implementation and to test other data structures, e.g. counting quotient filter [42], to decrease the memory requirements.

References

[1] S. S. Mande, M. H. Mohammed, and T. S. Ghosh, "Classification of metagenomic sequences: methods and challenges," *Briefings in Bioinformatics*, vol. 13, no. 6, pp. 669–681, Nov. 2012.

[2] J. Qian and M. Comin, "Metacon: Unsupervised clustering of metagenomic contigs with probabilistic k-mers statistics and coverage," *BMC Bioinformatics*, vol. 20, no. 367, 2019.

[3] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of Molecular Biology*, vol. 215, no. 3, pp. 403 – 410, 1990.

[4] Z. Zhang, S. Schwartz, L. Wagner, and W. Miller, "A greedy algorithm for aligning dna sequences," *Journal of Computational Biology*, vol. 7, no. 1-2, pp. 203–214, 2004.

[5] D. H. Huson, A. F. Auch, J. Qi, and S. C. Schuster, "Megan analysis of metagenomic data," *Genome Res.*, vol. 17, 2007.

[6] N. Segata, L. Waldron, A. Ballarini, V. Narasimhan, O. Jousson, and C. Huttenhower, "Metagenomic microbial community profiling using unique clade-specific marker genes," *Nat Methods*, vol. 9, 2012.

[7] A. Darling, G. Jospin, E. Lowe, F. I. Matsen, H. Bik, and J. Eisen, "Phylosift: phylogenetic analysis of genomes and metagenomes," *PeerJ*, vol. 2, 2014.

[8] S. Lindgreen, K. Adair, and P. Gardner, "An evaluation of the accuracy and speed of metagenome analysis tools," *Cold Spring Harbor Laboratory Press*, 2015.

[9] D. Wood and S. Salzberg, "Kraken: ultrafast metagenomic sequence classification using exact alignments," *Genome Biol.*, vol. 15, 2014.

[10] R. Ounit, S. Wanamaker, T. J. Close, and S. Lonardi, "Clark: fast and accurate classification of metagenomic and genomic sequences using discriminative k-mers," *BMC Genomics*, vol. 16, no. 1, pp. 1–13, 2015.

[11] S. Girotto, M. Comin, and C. Pizzi, "Higher recall in metagenomic sequence classification exploiting overlapping reads," *BMC Genomics*, vol. 18, no. 10, p. 917, Dec 2017.

[12] D. Kim, L. Song, F. Breitwieser, and S. Salzberg, "Centrifuge: Rapid and sensitive classification of metagenomic sequences," *Genome Research*, vol. 26, p. gr.210641.116, 10 2016.

[13] D. E. Wood, J. Lu, and B. Langmead, "Improved metagenomic analysis with kraken 2," *Genome biology*, vol. 20, no. 1, p. 257, 2019.

[14] J. Qian, D. Marchiori, and M. Comin, "Fast and sensitive classification of short metagenomic reads with skraken," in *Biomedical Engineering Systems and Technologies*, N. Peixoto, M. Silveira, H. H. Ali, C. Maciel, and E. L. van den Broek, Eds. Cham: Springer International Publishing, 2018, pp. 212–226.

[15] D. Marchiori and M. Comin, "Skraken: Fast and sensitive classification of short metagenomic reads based on filtering uninformative k-mers," in *BIOINFORMATICS 2017 - 8th International Conference on Bioinformatics Models, Methods and Algorithms, Proceedings; Part of 10th International Joint Conference on Biomedical Engineering Systems and Technologies, BIOSTEC 2017*, vol. 3, 2017, pp. 59–67.

[16] K. Břinda, M. Sykulski, and G. Kucherov, "Spaced seeds improve k-mer-based metagenomic classification," *Bioinformatics*, vol. 31, no. 22, p. 3584, 2015.

[17] J. A. Eisen, "Environmental shotgun sequencing: its potential and challenges for studying the hidden world of microbes," *PLoS Biol.*, vol. 5, 2007.

[18] D. Storato and M. Comin, "Improving metagenomic classification using discriminative k-mers from sequencing data," in *Bioinformatics Research and Applications*, Z. Cai, I. Mandoiu, G. Narasimhan, P. Skums, and X. Guo, Eds. Cham: Springer International Publishing, 2020, pp. 68–81.

[19] G. Marçais, B. Solomon, R. Patro, and C. Kingsford, "Sketching and sublinear data structures in genomics," *Annual Review of Biomedical Data Science*, vol. 2, no. 1, pp. 93–118, 2019. [Online]. Available: https://doi.org/10.1146/annurev-biodatasci-072018-021156

[20] S. Vinga and J. Almeida, "Alignment-free sequence comparison–a review," *Bioinformatics.*, vol. 19, 2003.

[21] M. Comin and D. Verzotto, "Whole-genome phylogeny by virtue of unic subwords," in *Database and Expert Systems Applications (DEXA), 2012 23rd International Workshop on*, Sept 2012, pp. 190–194.

[22] G. E. Sims, S. . R. Jun, G. A. Wu, and S. . H. Kim, "Alignment-free genome comparison with feature frequency profiles (ffp) and optimal resolutions," *Proc Nat Acad Sci.*, vol. 106, 2009.

[23] M. Antonello and M. Comin, "Fast alignment-free comparison for regulatory sequences using multiple resolution entropic profiles," in *Proceedings of the International Conference on Bioinformatics Models, Methods and Algorithms (BIOSTEC 2015)*, 2015, pp. 171–177.

[24] M. Comin and M. Antonello, "On the comparison of regulatory sequences with multiple resolution entropic profiles," *BMC Bioinformatics*, vol. 17, no. 1, p. 130, Mar 2016.

[25] M. Comin and D. Verzotto, "Beyond fixed-resolution alignment-free measures for mammalian enhancers sequence comparison," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 11, no. 4, pp. 628–637, July 2014.

[26] J. Goke, M. H. Schulz, J. Lasserre, and M. Vingron, "Estimation of pairwise sequence similarity of mammalian enhancers with word neighbourhood counts," *Bioinformatics*, vol. 28, no. 5, pp. 656–663, 2012.

[27] M. R. Kantorovitz, G. E. Robinson, and S. Sinha, "A statistical method for alignment-free comparison of regulatory sequences," *Bioinformatics.*, vol. 23, 2007.

[28] M. Antonello and M. Comin, *Fast Computation of Entropic Profiles for the Detection of Conservation in Genomes*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 277–288.

[29] ——, "Fast entropic profiler: An information theoretic approach for the discovery of patterns in genomes," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 11, no. 3, pp. 500–509, May 2014.

[30] Y. Shibuya and M. Comin, "Better quality score compression through sequence-based quality smoothing," *BMC Bioinformatics*, vol. 20, no. 302, 2019.

[31] ——, "Indexing k-mers in linear-space for quality value compression," *Journal of bioinformatics and computational biology*, vol. 7, no. 5, pp. 21–29, 2019.

[32] Y. W. Yu, D. Yorukoglu, J. Peng, and B. Berger, "Quality score compression improves genotyping accuracy," *Nature Biotechnology*, vol. 33, no. 3, pp. 240–243, Mar. 2015.

[33] M. Schimd and M. Comin, "Fast comparison of genomic and metagenomic reads with alignment-free measures based on quality values," *BMC Medical Genomics*, vol. 9, no. 1, pp. 41–50, 2016.

[34] M. Comin, A. Leoni, and M. Schimd, "Clustering of reads with alignment-free measures and quality values," *Algorithms for Molecular Biology*, vol. 10, no. 1, pp. 1–10, 2015.

[35] M. Comin and M. Schimd, "Assembly-free genome comparison based on next-generation sequencing reads and variable length patterns," *BMC Bioinformatics*, vol. 15, no. 9, pp. 1–10, 2014.

[36] B. D. Ondov, T. J. Treangen, P. Melsted, A. B. Mallonee, N. H. Bergman, S. Koren, and A. M. Phillippy, "Mash: fast genome and metagenome distance estimation using minhash," *Genome Biology*, vol. 17, no. 132, 2016.

[37] F. Breitwieser, D. Baker, and S. L. Salzberg, "Krakenuniq: confident and fast metagenomics classification using unique k-mer counts," *Genome biology*, vol. 19, no. 1, p. 198, 2018.

[38] M. Holtgrewe, "Mason: a read simulator for second generation sequencing data," 2010.

[39] A. Sczyrba, P. Hofmann, and A. C. McHardy, "Critical assessment of metagenome interpretation—a benchmark of metagenomics software," *Nature Methods*, vol. 14, p. 1063–1071, 2017.

[40] S. Girotto, C. Pizzi, and M. Comin, "Metaprob: accurate metagenomic reads binning based on probabilistic sequence signatures," *Bioinformatics*, vol. 32, no. 17, pp. i567–i575, 2016.

[41] A. Sobih, A. I. Tomescu, and V. Mäkinen, "Metaflow: Metagenomic profiling based on whole-genome coverage analysis with min-cost flows," in *Research in Computational Molecular Biology*, M. Singh, Ed. Cham: Springer International Publishing, 2016, pp. 111–121.

[42] P. Pandey, M. A. Bender, R. Johnson, and R. Patro, "A general-purpose counting filter: Making every bit count," in *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, 2017, pp. 775–787.

Davide Storato received the bachelor's degree and master degree in computer science, in 2017 and 2019, from the University of Padova, Italy. After graduation he has been a research fellow at the Department of Molecular Medicine, University of Padova, Italy.

Matteo Comin received the MS and PhD degrees in computer science from the University of Padova, Italy, in 2003 and 2007, respectively. His research interests focus on the area of algorithms for computational biology. He is interested in compression methods, and also in developing algorithms for next-generation sequencing. During his activity he has been a research intern at IBM T.J. Watson Research Center twice where he developed motif discovery systems for biological sequences. He has been a visiting researcher at the University of Purdue and at the Universitat Politcnica de Catalunya, Barcelona, Spain three times. He is a co-inventor of three US patent and author of more than sixty publications. In 2007 he received C. Offelli Award for best young researcher from the University of Padova. Since 2015 he is an Associate Professor at the University of Padova.