Kraken: ultrafast metagenomic sequence classification using exact alignments

Rushali Chopra

15.12.2023

1 Abstract

Kraken is a cutting-edge software tool, that employs a minimizer-based database structure to assign taxonomic labels to metagenomic DNA sequences efficiently. Kraken has demonstrated exceptional classification speed with a rate of over 4.1 million reads per minute being almost 900 times faster than Megablast. Compared to other established tools like MEGAN and MetaPhlAn, Kraken's robust performance and exact k-mer alignment approach makes it a significant tool for high-throughput metagenomic analysis. This report also delves into the intricacies of Kraken's workflow, emphasizing the role of k-mers, minimizers, and the minimizer offset array in streamlining database queries. Its applications in various research domains such as clinical diagnostics as well as recent advancements in its algorithm and database will also be discussed.

2 Introduction

Traditional microbiological methods relied on culturing individual microbes but they were not sufficient enough to capture the vast diversity of microbial ecosystems. The emergence of metagenomics was driven by the need to address the difficulties associated with culturing or isolating microorganisms and to encode a reservoir of undiscovered enzymes and metabolic functionalities.[11] Metagenomic analysis has revolutionized our understanding of microbial communities by enabling the study of genomic sequences directly from diverse environments. Thanks to metagenomics, researchers can now understand and analyze microbial life without isolating or culturing individual microbes. However, this poses significant computational challenges due to the sheer volume of data and the need for accurate taxonomic classification. Many species exhibit numerous similarities to already known organisms, which could be identified by applying alignment algorithms.[13] BLAST is one of the most prominent algorithms being used despite not being initially designed for metagenomic sequences.[1] Many other methods of classifying sequences such as MEGAN, PhymmBL, and NBC also exist. However, their processing burden is so demanding that a faster approach is required. Moreover, they are slower than BLAST.

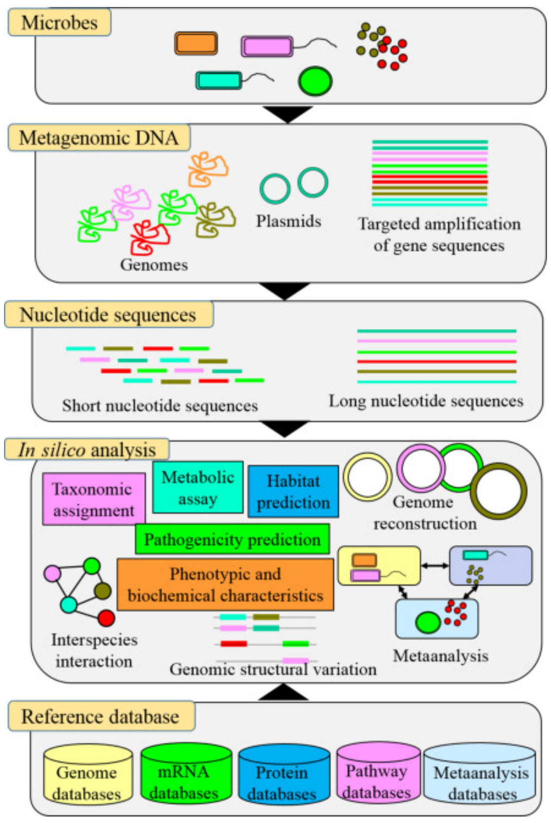

Figure 1: Metagenomics Workflow: It involves 1. Collecting microbes directly from environment, 2. DNA extraction and Library preparation, 3. Sequencing, and 4. Metagenomic sequence data analysis. Source: [8]

Abundance estimation programs comparatively work faster by creating smaller databases, thus eliminating the need to identify every read in a dataset. This is possible because of the presence of marker genes in their databases. Since the marker genes form a very small part of the genome sample, it is nearly impossible to get a detailed analysis of the sample or rather provide labels to every single read. Kraken, on the other hand, has proven to be faster than both classifiers and abundance estimation programs and more accurate than the best sequence classifiers. The algorithm being used by Kraken helps it to achieve sensitivity and precision at the genus level, which closely matches the efficiency of the fastest BLAST program namely Megablast.[13]

3 Methodology

3.1 Sequence Classification Algorithm

The very first step is collecting all k-mers present in sequence (S), which is to be classified, and denoting it as set K(S). Then every k-mer is mapped to the lowest common ancestor(LCA) taxon of all the genomes associated with that specific k-mer. The classification tree, which includes all these LCA taxons and also their ancestors in the taxonomy tree, is used for classifying a particular sequence (S). Each node in the classification tree has a weight that corresponds to the number of k-mers in K(S) that is mapped to the taxon linked with that node. The algorithm traverses from the root node to a leaf node along a specific path and subsequently evaluates each root-to-leaf (RTL) path by summing up all the weights along the path. The RTL path with the highest weight is identified as the most likely classification path, and sequence (S) is assigned a taxonomic label associated with the leaf node of this path(Orange node shown in Figure 2). If there are multiple maximum scores achieved by multiple RTL paths, the LCA of all the corresponding leaf nodes is selected as the final classification.

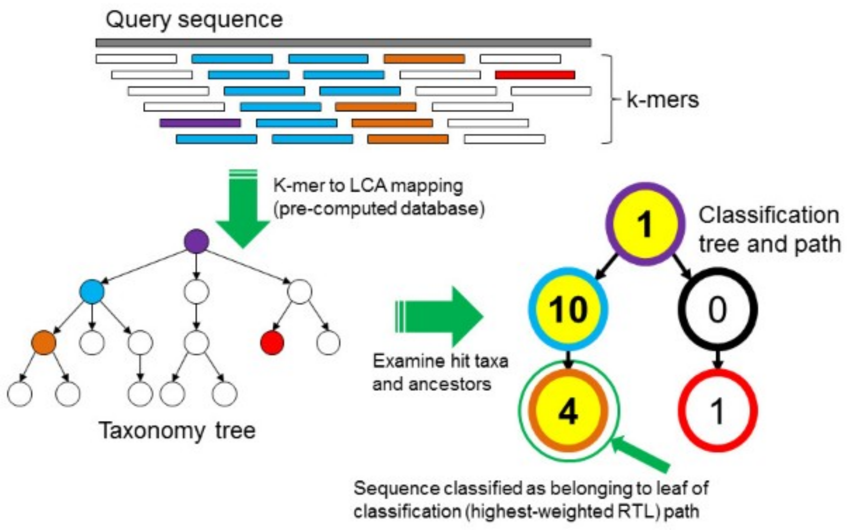

Figure 2: Kraken's classification algorithm

This summation of weights helps the algorithm to consider the individual contributions of k-mers associated with each node along the path. It further represents the strength of the association between the classified sequence and the taxonomic labels associated with those nodes. It also mitigates discrepancies during classification, particularly in cases of wrong taxonomic labels by certain k-mers. It also aims to compensate for discrepancies between the classified sequences and those present in the genomic library. These discrepancies are because of the variations exhibited by sequences in the genomic library due to genetic diversity, mutations, or sometimes sequencing errors. This incorporation of RTL path scoring significantly boosts accuracy and overall precision during classification.
In case when k-mers do not have a match in the genomic sequences database, those sequences are considered to be unclassified by the algorithm. Such sequences sometimes could impact the classification decision as they might produce divergent or inconclusive evidence regarding the sequence's taxonomic origin. It also highlights the limitations of the reference database indicating that certain sequences are not present in the dataset. Due to this, the algorithm fails to assign correct labels, leading to uncertainties in classification outcomes.

3.2 Database Query

For the efficient implementation of Kraken's algorithm, k-mers are mapped to taxa by querying the database, which is already set. Kraken, by default, has a genomic sequence library that is primarily based on completed microbial genomes in the National Center for Biotechnology Information's (NCBI) RefSeq database.[9] The next step is the usage of a jellyfish multithreaded k-mer counter for generating a database encompassing unique 31-mers from the library. Kraken not only uses jellyfish for creating and storing k-mers but rather stores the taxonomic ID numbers corresponding to the k-mers LCA values after the database creation is completed.[6] Then the processing starts, in which for every sequence, its associated taxon is utilized to set the stored LCA values for all k-mers belonging to that specific sequence. This means that the taxonomic information obtained from every sequence is essential for accurate sequence classification and for finding out the hierarchy to which k-mers belong. Furthermore, by effectively associating the corresponding LCA values with k-mers, Kraken can precisely map k-mers to their respective taxonomic positions in the classification tree. If in case the value for k-mer is already set, Kraken calculates the LCA of the stored value and taxon of the present sequence. This calculation is done by traversing the classification tree to find the shared ancestor that is common to both the existing LCA value and the current sequence's taxon and then identifying the lowest node in that tree that is an ancestor to both these values. Kraken has a very sophisticated strategy to enhance the efficiency of its database query process. It performs grouping of similar k-mers together implementing the minimizer concept, especially when querying adjacent k-mers that share a significant amount of sequence.

3.2.1 Minimizer Concept

The minimizer concept involves selecting a representative subset of k-mers, called minimizers, from a sequence in such a way that different sequences sharing a long enough subsequence will eventually select the same minimizer. This selection makes sure that only a small fraction of all possible k-mers in a sequence are stored as minimizers, thereby reducing storage requirements.[10]
The points listed below will explain how Kraken applies the minimizer concept:-

- *Canonical representation*:
  A DNA sequence (S) is canonically represented as the lexicographically smaller of S and its reverse complement. The reverse complement is obtained in which a sequence (S) is reversed and every nucleotide is replaced by its complementary base (A with T, C with G, and vice versa). After comparison between the original sequence and its reverse complement, the lexicographically smaller of the two sequences is selected. This step particularly eliminates ambiguity and is a consistent way to represent sequences. It further simplifies the process of identifying minimizers within DNA sequences.

- *Minimizers calculation and grouping:*
  This step involves finding out the k-mer's minimizers of length M by considering all the canonical representations of M-mers within k-mer. Lexicographically, the smallest minimizer of all is identified as the representative k-mer in a group of similar k-mers. These representative k-mers are organized by kraken consecutively in the

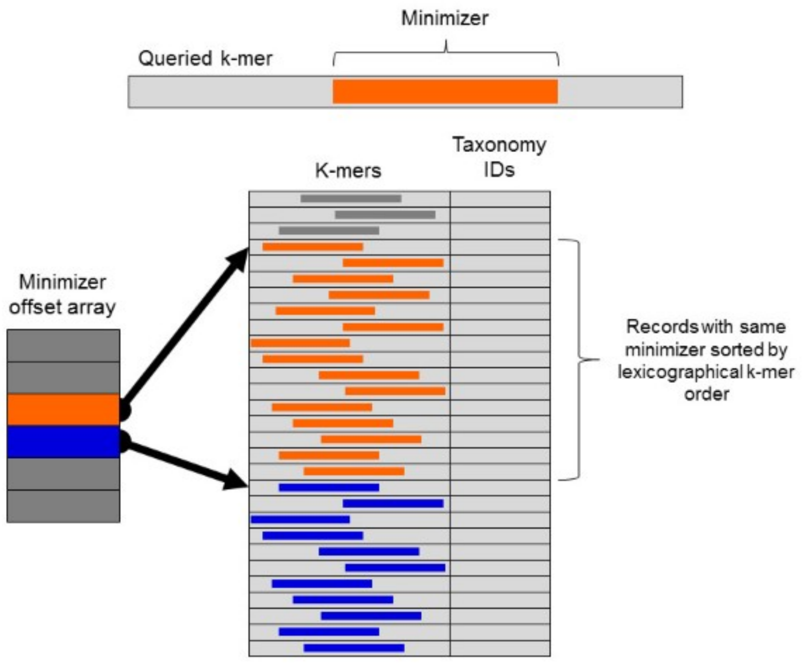
Minimizer
Queried k-mer
Taxonomy IDs
K-mers
Minimizer offset array
Records with same minimizer sorted by lexicographical k-mer order

Figure 3: Database Search: K-mers to be queried are looked up in Minimizer offset array

database. These are then sorted in lexicographical order based on their canonical representations. This facilitates efficient data retrieval, thus enhancing Kraken's speed and accuracy in classification.

- *Database Search process:*
  - Minimizer Offset Array:
    It is a data structure used in Kraken's database to identify the positions in the database that contain k-mers with the same minimizer as the queried k-mer (shown in orange in Figure 3) and the next potential minimizer (shown in blue in Figure 3). By referencing this array, Kraken only targets the specific range of records relevant to the query k-mer's minimizer thereby optimizing the search process.
  - Lexicographical sorting and Binary search:
    After using a Minimizer offset array to identify a specific range of relevant records, Kraken sorts the k-mers lexicographically. These k-mers are also linked to their respective taxonomic IDs as shown in Figure 3. A binary search is then performed that quickly identifies the target k-mer within the sorted range of k-mers, retrieves its taxonomic ID, and streamlines the classification process.

3.2.2 Search Algorithm enhancements

Since the minimizers are normally shared among adjacent k-mers, the search range for a query is constant between consecutive queries. At the time of the first query, data associated with it is already brought into the CPU cache rather than the RAM; which subsequently escalates the upcoming queries to assess the data from cache and not RAM.

This strategy minimizes redundant computations by avoiding recalculating the minimizer for each query. Additionally, kraken first performs a search in its previous query range and if the queried k-mer is found, the query is concluded promptly. If this is not the case, computation of the minimizer is carried out. The query fails if this new minimizer has a match with the minimizer of the last queried k-mer. Kraken initiates a new search only if the minimizer changes. This helps kraken reduce computational complexities and expedite the process of query processing. Some things were also optimized to increase the efficiency of the search process, specifically the usage of exclusive-or (XOR) operation.[13] It was emphasized by Roberts et al. [10] that when M-mers are ordered lexicographically, they have a skewed distribution of minimizers. The most probable reason behind this could be M-mers, which are not as complex as others. This subsequently escalates the time required for the search range process. The XOR operation involves halving the bits of each M-mer's canonical representation, thus speeding the search process. By scrambling the standard lexicographical ordering of M-mers using XOR operation, Kraken creates a more uniform distribution of minimizers thereby contributing to faster and more accurate query processing.

4 Dataset selection and overview

4.1 Data Source

The dataset used was constructed from simulated metagenomic reads by amalgamating real sequences, which were derived from projects that sequenced microbial genomes. Bacterial whole-genome shotgun sequences found in the GAGE-B project [5] and the NCBI Sequence Read Archive [9] were the main sources for the reads obtained.

4.2 Dataset size and description

Three simulated metagenomes were created: the HiSeq metagenome, the MiSeq metagenome and the simBA-5 metagenome. The HiSeq and MiSeq metagenomes had 20 sets of bacterial whole-genome shotgun reads, where each metagenome consisted of sequences from ten distinct genomes with 10% of the sequences in both the 10,000 and 10 million read samples to maintain consistency in the sequence abundance. The simBA-5 metagenome featured simulated bacterial and archaeal reads present in the RefSeq database.

4.3 Data collection and preprocessing:

Illumina HiSeq and MiSeq sequencing platforms were used to sequence the microbial genomes to select the reads, whereas for simBA-5 reads Mason read simulator [2] was used. Before conducting the analysis, all sequences were trimmed to avoid any low-quality bases or adapter sequences ensuring no bias or error in the analysis. The simBA-5 metagenome was created with specific SNP and Indel rates of 0.1%, with an error rate of about five times more than expected.

4.4 Purpose

The decision to employ simulated metagenomes was made because of the constraints associated with genuine metagenomic reads while evaluating classification accuracy. The real metagenomic reads are often known to contain unknown species that cannot be mapped to already existing reference genomes.[15] The advantage of using simulated metagenomic datasets is a controlled environment where microbial composition, sampling of sequences, and other factors like noise are known.[7] They are also used for benchmarking and studying how multiple factors affect the analysis of the sequencing data.[4]

5 Results and Discussion

Classification accuracy and speed were assessed for each of the three metagenomes by comparisons between five distinct classification programs namely Kraken, Megablast, PhymmBL, NBC, and MetaPhlAn. The evaluation also included different variants of Kraken such as MiniKraken, Kraken-Q, MiniKraken-Q, and KrakenGB. PhymmBL65 was a selective classifier that was created by setting a lower bound of 0.65 for confidence at the genus-level. This meant that only classifications that fall under a category of confidence score equal to or higher than 0.65 would be considered for future analyses. This threshold certainly acts as a threshold to improve the quality of results obtained while classification.[13]

5.1 Classification accuracy

Some genomes often lack their taxonomic information across all ranks, due to which accuracy for kraken was primarily measured at the genus level. Genus-level sensitivityA/B is defined as the ratio of correctly classified reads (A) at genus rank to the total number of reads (B) with any genus that is assigned to it, whereas precision at a particular taxonomic rank $C/D+E$ is articulated as the ratio of the number of reads labeled at or below the correct taxon at the measured rank (C) to the sum of the number of reads labeled at or below the rank measured (D) and number of reads labeled falsely above the rank measured (E). For the comparison in the context of classification accuracy, 10,000 sequences from each of the three simulated metagenomes were classified. Surprisingly, PhymmBL and NBC could classify all sequences as possible which led to more false positives whereas Kraken and Megablast had left some sequences as unclassified due to insufficient information. That is also the reason why Kraken tends to have a higher precision than sensitivity. The sensitivity and precision levels demonstrated by Kraken are comparable to those of Megablast.[13]

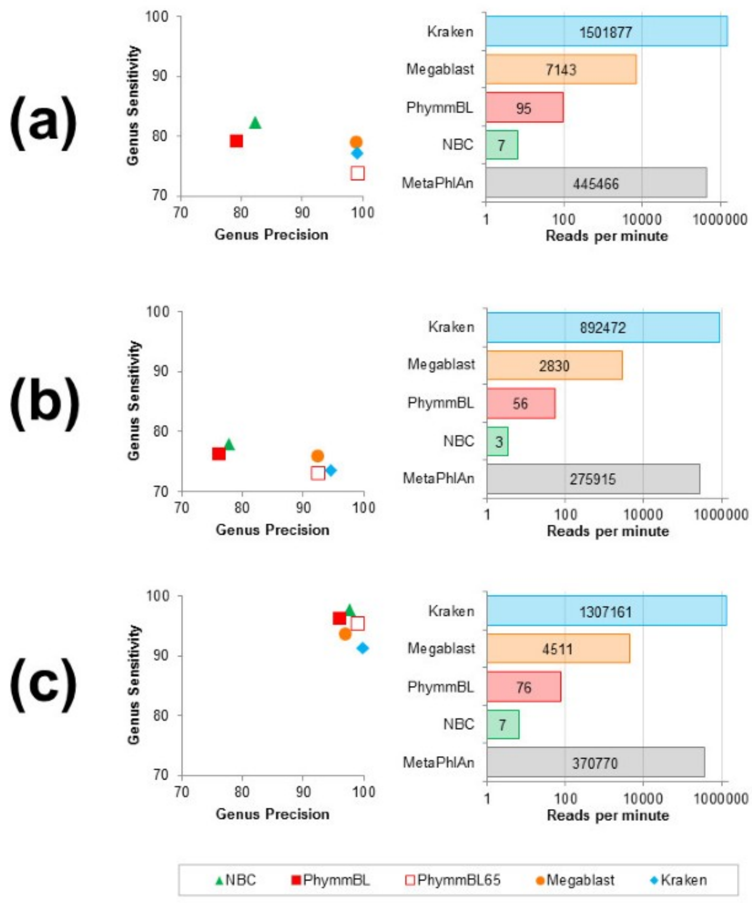

Figure 4: Comparison of classification speed and accuracy of classification programs for three simulated metagenomes. Results shown in a) represent the HiSeq metagenome, b) represent the MiSeq metagenome, and c) represent the simBA-5 metagenome.

As depicted in the Figure 4 and Figure 5, Kraken's precision was almost more than 90% for all three metagenomes but it showed a significant increase in its sensitivity in simBA-5 metagenome dataset. This could be attributed to Kraken's robust classification approach which enables it to handle high sequence error rates effectively. In fact, all other classifiers also showed higher sensitivity in simBA-5 metagenome than the other two metagenomes. LMAT, another classifier, utilizes a k-mer indexing scheme which is similar to the approach used by Kraken. Kraken's performance was analyzed on a dataset that was initially used to obtain LMAT's results. Kraken outperformed LMAT in the identification of the reads' origin and detection of the presence of the species within the sample. In the context of accuracy as well, Kraken showed a high accuracy rate of 89% whereas LMAT achieved only 74%. This could be due to the dataset used as it was simulated without errors and had genomes present in both Kraken's and LMAT's databases.

5.2 Classification speed

For evaluating classification speed, datasets had 10,000 reads each for all the programs but Kraken and MetaPhlAn had 10,000,000 reads each. The difference in reads used was for reducing the effect of operations such as loading database files and to ensure the statistical stability in the performance metrics obtained. The data processing rate of Kraken was between 8,90,000 to 1.5 million reads per minute (rpm) across the three metagenomes. On the contrary, PhymmBL, NBC, and Megablast demonstrated very low speed with rates below 100 rpm for PhymmBL and below 10 rpm for NBC.(Figure 4) As the read length increased in MiSeq metagenome, the speed was decreased by 1.6 times in Kraken and MetaPhlAn. Kraken also showed faster processing speeds being 38.82 times faster than LMAT and 7.55 times faster than LMAT's other version. Kraken's exceptional speed advantage, being 150 to 240 times faster than Megablast across all three metagenomes, shows its unmatched efficiency in sequence classification.

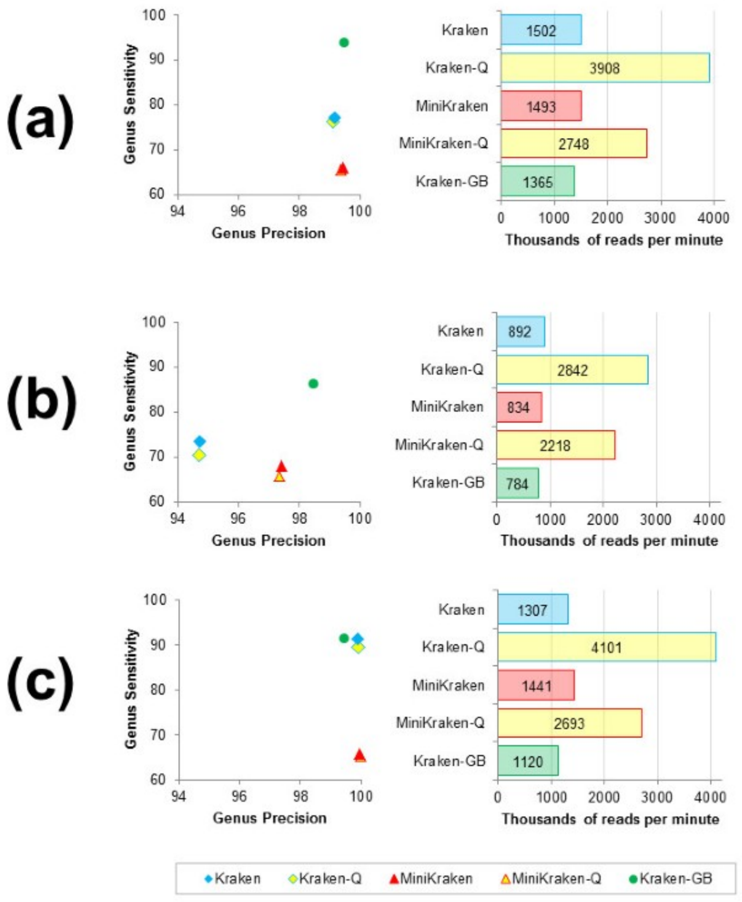

Figure 5: Comparison of classification speed and accuracy of other variants of kraken for three simulated metagenomes Results shown in a) represent the HiSeq metagenome, b) represent the MiSeq metagenome, and c) represent the simBA-5 metagenome.

The RAM requirements of Kraken to hold its entire database led to its smaller database version, called MiniKraken, which reduced its memory requirements. MiniKraken showed high precision across all three metagenomes which eliminated the need to match all the k-mers in the database. This approach, in which rather than matching all the k-mers, the query is supposed to match the first k-mer with the k-mer in the database, and then use its corresponding LCA for classification, is used in Kraken-Q and MiniKraken-Q.On the high-error simBA-5 metagenome, MiniKraken's sensitivity was more than 25 percentage points lower than Kraken's, indicating that for short reads, high error rates can cause substantial loss in sensitivity. However, for all three metagenomes, MiniKraken was more precise than Kraken. Kraken-GB contains genomes from GenBank's draft and completed genomes for bacteria and archaea. Due to its larger database size and two genomes in metagenomic samples, it demonstrated higher sensitivity than Kraken. The overall performance of Kraken-Q was better than Kraken's across all three metagenomes. Kraken-GB was not even as fast as half the speed of Kraken-Q (Figure 5).

5.3 Other Experiments

Kraken was also tested to simulate the presence of novel organisms, in which the main challenge was to assign correct species labels. For the simBA-5 metagenome, organisms belonging to the same clade as of reads were excluded from the Kraken database. This was done to mimic an environment where Kraken would encounter the taxonomic groups not represented in the database. Kraken, however, showed high precision at the rank level but relatively very low sensitivity values of about 33%. Kraken was also utilized to analyze three saliva samples belonging to the Human Microbiome Project.[3] For this, Kraken's database was specifically modified to include bacterial, viral, and human genomes for classification. 62.8% of reads were unclassified because they were totally unknown or were faced with divergent or novel.

6 Conclusion

In conclusion, the evaluation of Kraken and other classification algorithms on simulated metagenomic datasets highlights Kraken's competitive accuracy and efficiency in taxonomic classification. Firstly, in terms of classification accuracy, Kraken demonstrated high precision levels, consistently exceeding 90% across all metagenomes. This precision was particularly noteworthy given Kraken's robust classification approach, enabling it to handle high sequence error rates effectively. Additionally, Kraken exhibited competitive sensitivity levels, especially evident in the simBA-5 metagenome dataset with elevated error rates. Secondly, regarding classification speed, Kraken demonstrated exceptional efficiency by employing strategies such as exact-match database k-mer queries. This notable speed advantage over other classifiers, including Megablast and LMAT, emphasized Kraken's efficiency in large-scale sequence classification tasks. Moreover, the introduction of database variants like MiniKraken and Kraken-GB provided insights into optimizing memory usage and expanding database coverage, further enhancing Kraken’s versatility and applicability in diverse metagenomic analyses. Despite krakenGB's high sensitivity, the issue of the presence of contaminant sequences is a challenge for maintaining database integrity. The comprehensive evaluation of Kraken and its variants reaffirms their significance as powerful tools for metagenomic sequence classification. With high precision, competitive sensitivity, and exceptional speed, Kraken offers researchers a reliable and efficient solution for analyzing complex microbial communities. By continuously refining its algorithms and database structures, Kraken remains at the forefront of metagenomic analysis, contributing to advancements in understanding microbial diversity and ecosystem dynamics.

7 Future Outlook

Kraken, besides being used for taxonomic classification, can also find applications in genome assembly programs, and de Bruijn graphs.[16] It is also used for the detection of pathogens in environmental samples. Because of Kraken's high precision, a two-step classification strategy could be used for maximizing sensitivity. By initially running Kraken to classify the reads, and then employing other classification programs for those reads that were left unclassified, high sensitivity could also be achieved. However, Kraken's performance is constrained by its memory usage.[13] Other than Kraken, there are other three major versions of the software existing today which are Kraken2, KrakenUniq, and Kraken2Uniq.

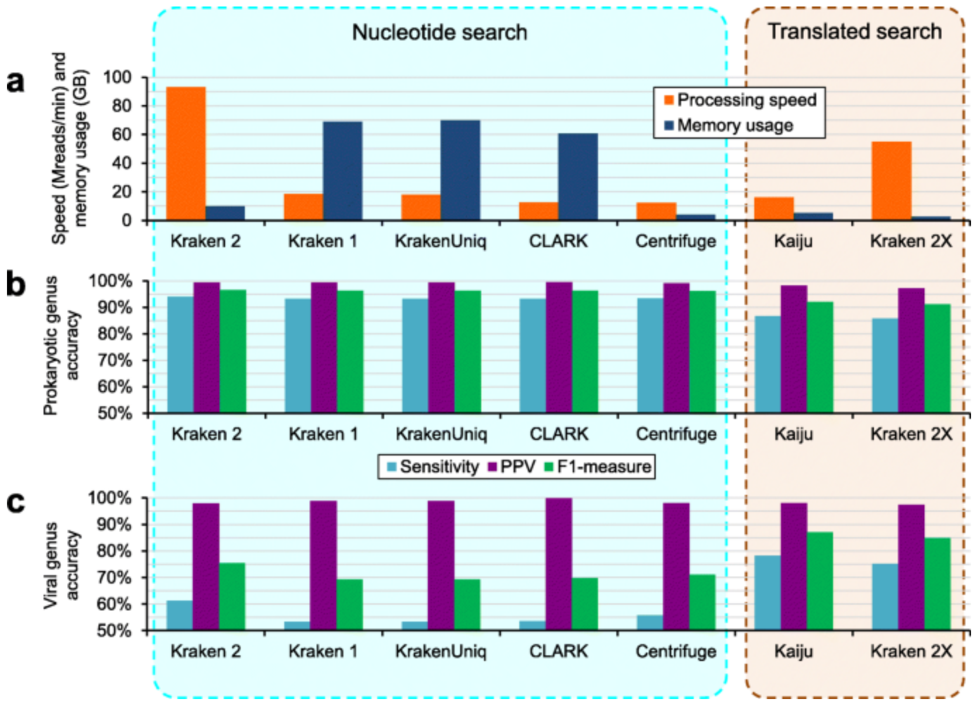

Figure 6: Comparison between various sequence classification tools

Kraken2 achieves an 85% reduction in memory requirements by changing its data structure. It also could classify paired-end data at over 93 million rpm using 16 threads, which is 5 times faster than Kraken. Data structures like counting quotient filter data could be integrated which will optimize Kraken2's performance in specific computing environments. Kraken2 combined with Bracken could help in the estimation of species-level abundances in metagenomic data, thereby helping researchers with the microbe analysis. By leveraging the KrakenUniq's HyperLogSketch technique, Kraken2 could potentially help more in disease diagnosis.[12]Since Kraken is known for its relatively fast classification speed, especially after deployment on servers with ample memory(>100 GB), MetaPhlAn2 could be the potential tool in case of constrained computational capabilities.[14]

References

[1] Arthur Brady and Steven L Salzberg. "Phymm and PhymmBL: metagenomic phylogenetic classification with interpolated Markov models". In: *Nature Methods* 6 (2009), pp. 673–676. DOI: 10.1038/nmeth.1358.

[2] Manuel Holtgrewe. *Mason*. http://www.seqan.de/projects/mason/. Accessed: 30.03.2024.

[3] Curtis Huttenhower et al. "Structure, function and diversity of the healthy human microbiome". In: *Nature* 486 (2012), pp. 207–214.

[4] Suyu Liu et al. "Comprehensive simulation of metagenomic sequencing data with non-uniform sampling distribution". In: *Quantitative Biology* 6 (2018), pp. 175–185. DOI: 10.1007/s40484-018-0142-9. URL: https://doi.org/10.1007/s40484-018-0142-9.

[5] Tanja Magoc et al. "GAGE-B: an evaluation of genome assemblers for bacterial organisms". In: *Bioinformatics* 29 (2013), pp. 1718–1725. DOI: 10.1093/bioinformatics/btt273.

[6] Guillaume MarÃ§ais and Carl Kingsford. "A fast, lock-free approach for efficient parallel counting of occurrences of k-mers". In: *Bioinformatics* 27.6 (Jan. 2011), pp. 764–770. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btr011. eprint: https://academic.oup.com/bioinformatics/article-pdf/27/6/764/48866141/bioinformatics_27_6_764.pdf. URL: https://doi.org/10.1093/bioinformatics/btr011.

[7] Konstantinos Mavromatis et al. "Use of simulated data sets to evaluate the fidelity of metagenomic processing methods". In: *Nature Methods* 4.6 (2007), pp. 495–500. DOI: 10.1038/nmeth1043. URL: https://doi.org/10.1038/nmeth1043.

[8] *Metagenomics*. ScienceDirect. Accessed on March 29, 2024. URL: https://www.sciencedirect.com/topics/immunology-and-microbiology/metagenomics.

[9] Kim D Pruitt et al. "NCBI Reference Sequences (RefSeq): current status, new features and genome annotation policy". In: *Nucleic Acids Research* 40 (2012), pp. D130–D135. DOI: 10.1093/nar/gkr1079.

[10] Michael Roberts et al. "Reducing storage requirements for biological sequence comparison". In: *Bioinformatics* 20.18 (2004), pp. 3363–3369.

[11] Carola Simon and Rolf Daniel. "Metagenomic analyses: past and future trends". In: *Applied and Environmental Microbiology* 77.4 (2011), pp. 1153–1161. DOI: 10.1128/AEM.02345-10.

[12] Derrick E Wood, Jessica Lu, and Ben Langmead. "Improved metagenomic analysis with Kraken 2". In: *Genome Biol* 20.1 (2019), p. 257. DOI: 10.1186/s13059-019-1891-0. URL: https://doi.org/10.1186/s13059-019-1891-0.

[13] Derrick E. Wood and Steven L. Salzberg. “Kraken: ultrafast metagenomic sequence classification using exact alignments”. In: *Genome Biology* 15.3 (2014). DOI: 10.1186/gb-2014-15-3-r46.

[14] Simon H. Ye et al. “Benchmarking Metagenomics Tools for Taxonomic Classification”. In: *Cell* 180.1 (2019), pp. 1–17. DOI: 10.1016/j.cell.2019.07.010.

[15] Xuegong Zhang et al. “Reading the Underlying Information From Massive Metagenomic Sequencing Data”. In: *Proceedings of the IEEE* PP (Sept. 2016), pp. 1–15. DOI: 10.1109/JPROC.2016.2604406.

[16] Aleksey V Zimin et al. “The MaSuRCA genome assembler”. In: *Bioinformatics* 29 (2013), pp. 2669–2677. DOI: 10.1093/bioinformatics/btt476.