CAMI II Challenge NBC++ Final Report

Group Members: Ritesh Chidambaram, Yiru Wang, Yupei Yang

Instructor: Dr. Gail Rosen

TA: Zhengqiao Zhao

I. Abstract

Metagenomics emerged as the genomic analysis of microbial DNA, and it is also defined as a significant research field for human health and disease. Metagenomic researches are shown on increasingly larger scales [1]. This is a critical task as only 1-2% of bacteria can be cultured in the laboratory and identifying the unknown species in environmental samples has proved to be a challenging task. The ability to identify unknown species without a-priori knowledge of what a sample contains would open new doors in microbial ecology, virology, microbiology and biomedical research.

In this project, we use a Naive Bayes Classifier implemented in C++ to perform the taxonomic classification. The tools of original NBC++ are used for training and classification of genomes data and the CAMI dataset is downloaded for testing and evaluating our algorithm [2]. Two scripts were developed for NBC++ that automatically parse a random input fastq file to the form that NBC can take as input, and automatically parse the NBC++ prediction and convert it to TXT format. At the same time, the comparison also performed between the Naïve Bayes Classifier and other published metagenomic data classifier. Kaiju, which finds maximum exact matches on the protein-level using the Burrows–Wheeler transform, classifies reads with higher

sensitivity and similar precision compared with current k-mer-based classifiers [1]; Kraken, using database containing a record of the LCA of the k-mer to build a user-specified genomic library that allows for quick searching for the most specific nodes in the classification tree to determine the appropriate markers for the sequence [3]. This classifies could meet our testing and comparison expectation.

II. Literature Review

Genomic sequence classification has become an increasingly popular field of study in the past decade.The most well-known such algorithm, and one of the best methods for assigning a taxonomic label to an unknown se- quence, is the BLAST program [4] which can classify a sequence by finding the best alignment to a large data-base of genomic sequences. Naïve Bayes Classifier (NBC) [5] applies a Bayesian rule to distributions of k-mers within a genome. In the MEGAN [6] program, a sequence is searched (using BLAST) against multiple databases, and the lowest common ancestor (LCA) of the best matches against each database is assigned to the sequence. PhymmBL [7,8] combines the results of BLAST with scores produced from interpolated Markov models to achieve higher accuracy than BLAST alone.

For this literature review, “Kraken: ultrafast metagenomic sequence classification using exact alignments” [2] and “Fast and sensitive taxonomic classification for metagenomics with Kaiju” [1] are used for comparison of the previous methods of taxonomic classifications.

The core of Kraken is a database containing a record of the lowest common ancestor (LCA) of the k-mer and all organisms whose genome contains the k-mer. The database is built using a user-specified genomic library that allows for quick searching for the most specific nodes in the classification tree associated with a given k-mer. The sequences are classified by querying the database of each k-mer in the sequence and then using the resulting LCA taxonomic group to determine the appropriate markers for the sequence. Sequences without k-mers in the database are retained by the Kraken as unclassified.

Figure 1 provided by “Kraken: ultrafast metagenomic sequence classification using exact alignments” shows the basic algorithm of classification. To classify a sequence, each k-mer in the sequence is mapped to the LCA of the genomes that contain that k-mer in a database. The taxa associated with the sequence’s k-mers, as well as the taxa’s ancestors, form a pruned subtree of the general taxonomy tree, which is used for classification. In the classification tree, each node has a weight equal to the number of k-mers in the sequence associated with the node’s taxon. Each root-to-leaf (RTL) path in the classification tree is scored by adding all weights in the path, and the maximal RTL path in the classification tree is the classification path (nodes highlighted in yellow). The leaf of this classification path (the orange, leftmost leaf in the classification tree) is the classification used for the query sequence.

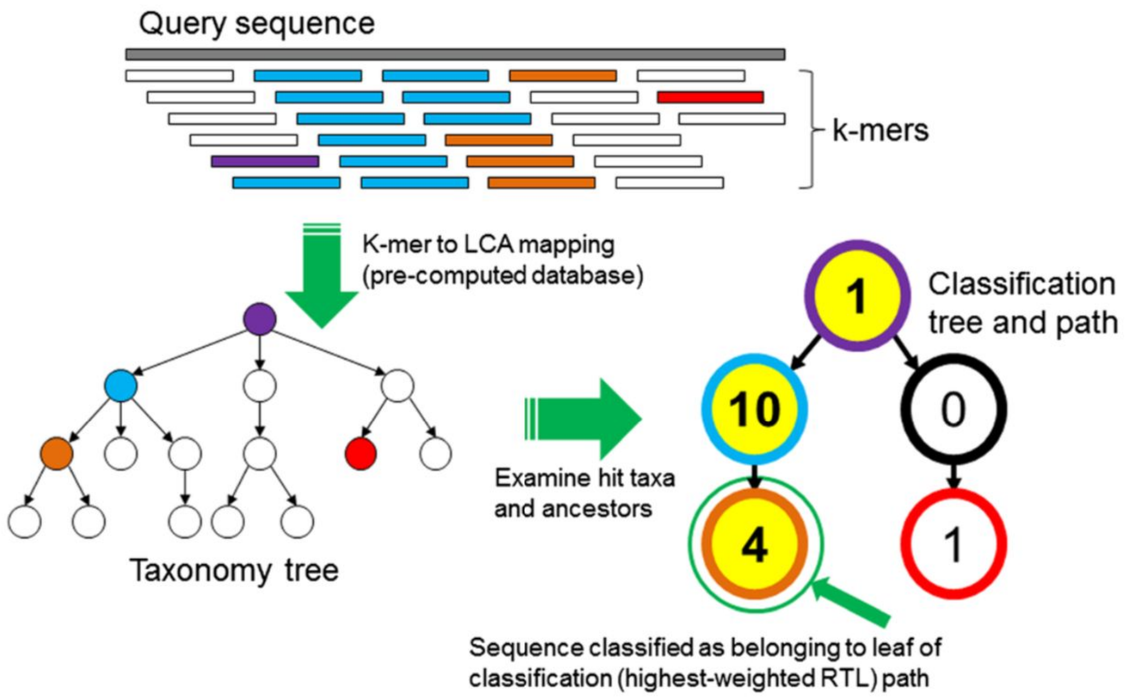


Figure 1: Kraken sequence classification algorithm.

The NBC++ classifier try to classify sequences as accurately as possible, however, kraken leave some sequences unclassified if insufficient evidence exists. To compare the Kraken's accuracy to other classifiers, the author of the Kraken classified 10,000 sequences from their simulated metagenomes and measured genus-level sensitivity and precision. Sensitivity, here, refers to the proportion of sequences assigned to the correct genus. Precision also known as positive predictive value, refers to the proportion of correct classifications, out of the total number of classifications attempted.

Classification speed is another significance that need to compare. The results shown in Figure 2 provided by “Kraken: ultrafast metagenomic sequence classification using exact alignments” gave the comparison of classification accuracy and speed between Kraken and other methods.

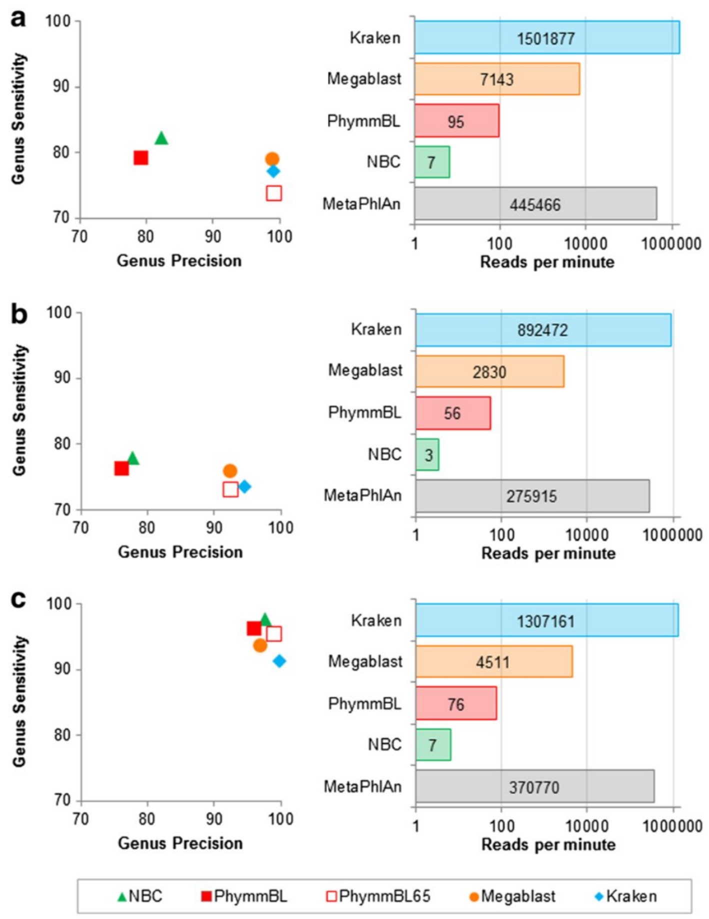


Figure 2. Classification accuracy and speed comparison

Different from Kraken, Kaiju performs protein-level sequence classification, translating metagenomic sequencing reads into the six possible reading frames and searching for maximum exact matches (MEMs) of amino acid sequences in a given database of annotated proteins from microbial reference genomes. Figure 3 provided by “Fast and sensitive taxonomic classification for metagenomics with Kaiju”shows the algorithms steps.

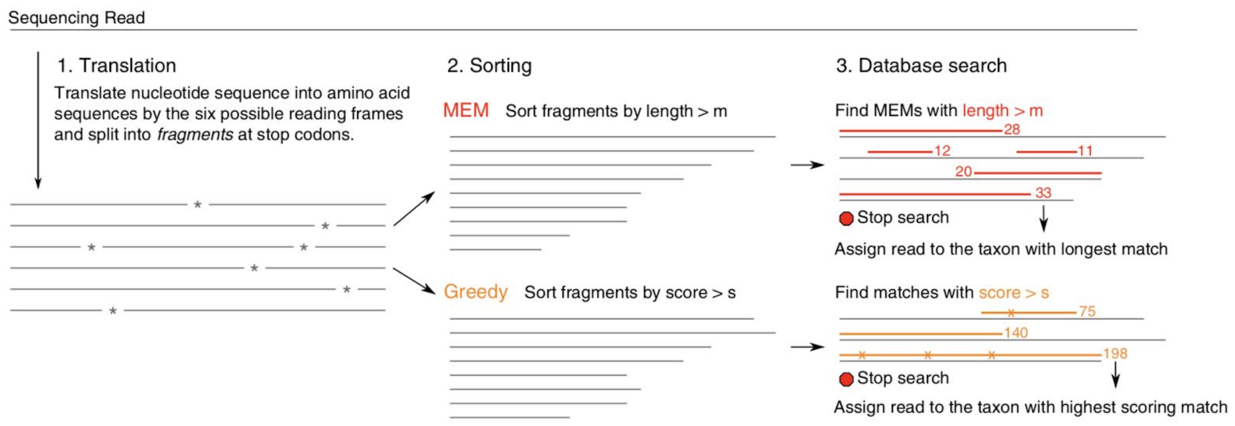

Sequencing Read
1. Translation
Translate nucleotide sequence into amino acid sequences by the six possible reading frames and split into *fragments* at stop codons.
2. Sorting
MEM Sort fragments by length > m
Greedy Sort fragments by score > s
3. Database search
Find MEMs with length > m
28
12
11
20
33
Stop search
Assign read to the taxon with longest match
Find matches with score > s
75
140
198
Stop search
Assign read to the taxon with highest scoring match

Figure 3. Algorithms of Kaiju

First of all, the sequencing reads are translated into six possible reading frames and the resulting amino acid sequence is divided into fragments at the stop codon. The segments are then sorted according to the length of the segment (MEM mode) or the BLOSUM62 score (greedy mode). The segment list of the classification is then searched against the reference protein database using a backward search algorithm on the Burrows–Wheeler transform (BWT). Although the MEM mode only allows for an exact match, the Greedy mode extends its left-end match by allowing substitution. Once the remaining segments in the list are shorter than the best match currently obtained (MEM) or a better score (greedy) is not obtained, the search stops and retrieves the classification identifier for the corresponding database sequence.

To compare the average of sensitivity and precision, the author provided the comparison of that between Kaiju and Kraken in different reads type shown in Figure 4. As that figure shown, Kaiju Greedy-1 and Greedy-5 have relatively highest precision and sensitivity both in genus and phylum, whereas Kraken has more “normal” performance.

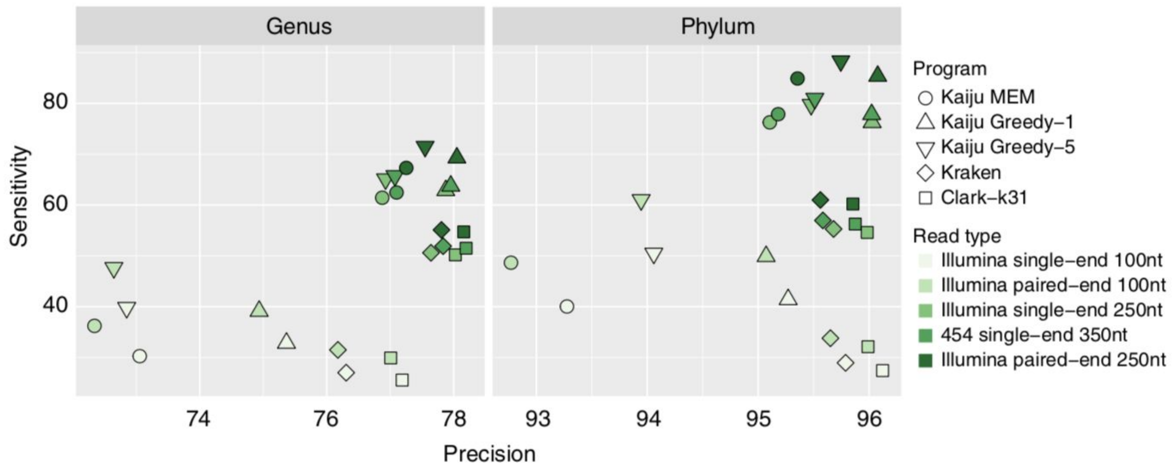


Figure 4. Average sensitivity and precision between Kaiju, Kraken, and Clark

Figure 5 shows the classification speed between Kaiju and Kraken in 5 different model testing, which was provided by "Fast and sensitive taxonomic classification for metagenomics with Kaiju". As figure shown, Kaiju MEM has the fastest speed than others, and Kraken also has relatively faster speed that other methods.

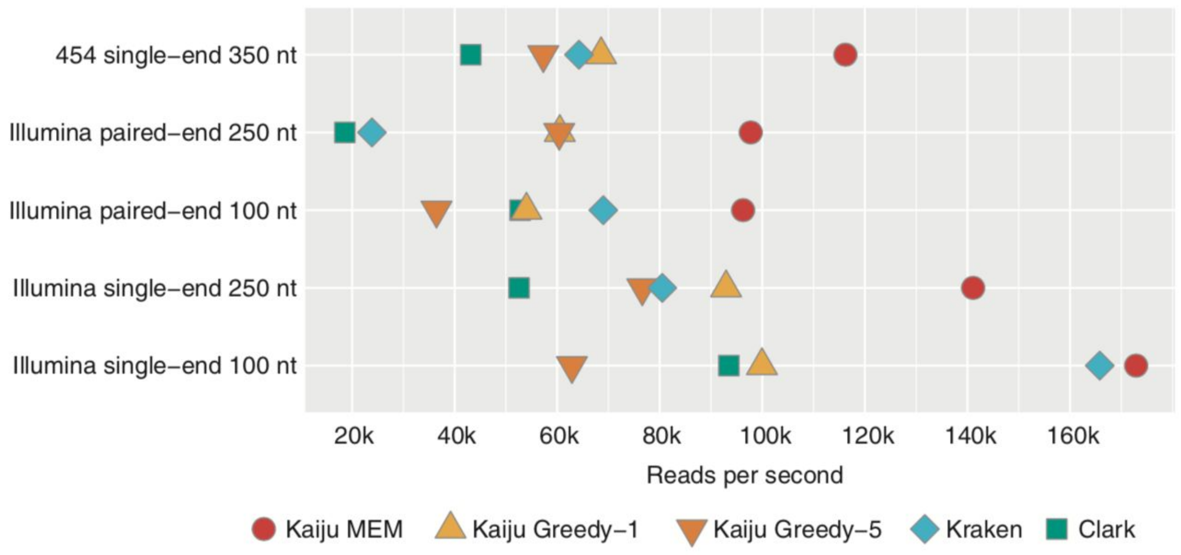
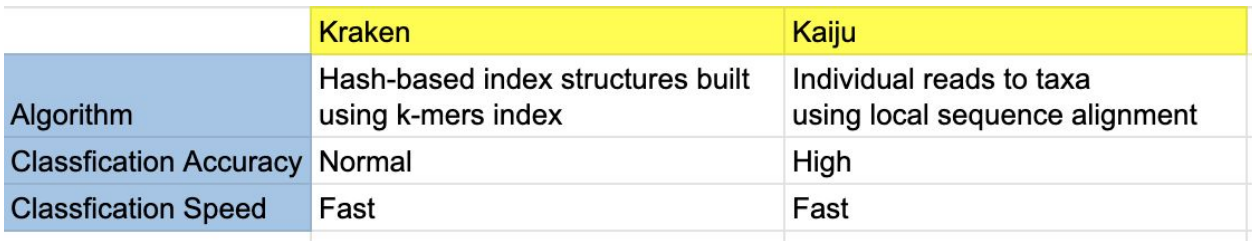


Figure 5. Classification speed between Kaiju, Kraken, and Clark

Table 1 shows the comparison between Kraken and Kaiju.

Table 1. Comparison of Kraken and Kaiju



	Kraken	Kaiju
Algorithm	Hash-based index structures built using k-mers index	Individual reads to taxa using local sequence alignment
Classfication Accuracy	Normal	High
Classfication Speed	Fast	Fast

For our CAIM profiling challenge project, we applied NBC++ [2] tools to predict strain madness datasets provided by CAMI which are simulated long read and short read shotgun metagenome data, including a large amount of strain-level variation. We would finally submit the results of prediction for this datasets to CAMI as our end-of-the-term deliverables. To complete this project, minimum 5 weeks need to take due to very large size of dataset. In the first week, the training model was built by using customized database. In the second week, the CAMI datasets were prepared so that NBC++ could take as inputs, such as conversion between fastq and fasta, split the reads into different file to increase testing speed, etc. In the third week, the prepared CAMI data were tested by our training model. In the fourth week, the testing results should be converted and concluded into human-readable format and CAMI acceptable format. In the final week, the prediction results and customized database should be package into a container to submit . One extra week may need due to the large size of datasets and relatively lower speed of testing.

III. Materials and Methods

Metagenomic tools are built to help perform this analysis to enable the population analysis of un-culturable or previously unknown microbes, but the interpretation of metagenomics relies on sophisticated computational approaches such as short-read assembly, binning and taxonomic classification. These initial data pre-processing methods are highly critical as all subsequent analysis relies on accurate data pre-processing. Although many classification programs now achieve high speed by comparing genomic k-mers, large fractions of the metagenomic reads remain unclassified since they often lack sensitivity for overcoming evolutionary divergence [1].

In this project, we used the Naïve Bayes Classifier to perform taxonomic classification to identify the different species present in an unknown sample. The Naive Bayes classifier for metagenomic data is created by Ecological and Evolutionary Signal-processing and Informatics Laboratory of Drexel University [2]. The training and classifying modes are supported. During training to create or add new sequences to an output file, based on the name of its respective class, the tags are expressed by placing each sequence into a subfolder of the source directory. And it is also useful when classifying to use existing output files and classify reads. If unsupervised runs are required, the current solution is to simply group all the reads into a folder with arbitrary tags. When running the NB.run script, different modes should be selected – training to create or add new sequences to a savefile and classifying to use existing savefiles and classifying reads [2]. For training NB classifier, the command should contain the path to data, which is used for training, and the total number of folders. For testing NB classifier, the command should contain the path to model, which is obtained by training, and the path to testing data.

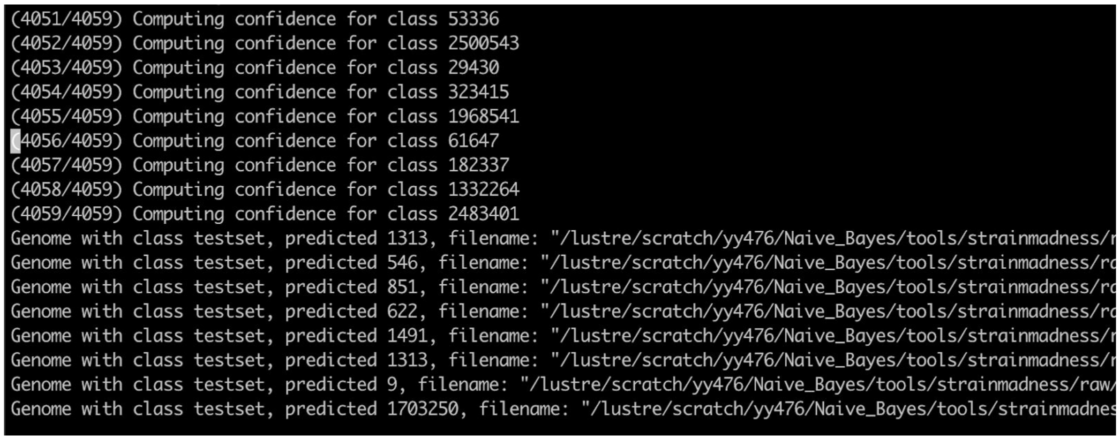
Before training, the download.sh should be run first to download the training dataset genome.fa from /mnt/HA/groups/rosenGrp/zz374/INBC_latest/genomes. When the download step is complete, we can start to run the kmr_counting.sh script to count the number of kmers in each genome before training. Running the train.sh script trains the NBC model for the testing step. Implementing this script should give input as path to dataset file downloaded before and change the output to a folder where the output model to be saved. And another parameter to set inside the code is the kmer count which usually defaulted by 15. After that, in order to parse the CAMI dataset as an input for NBC++ to test, there are several processes used to convert the CAMI datasets into the format which NBC++ need. When the downloading bash script was implemented in proteus to obtain the CAMI dataset, we would finally get many compressed fastq files. The first step to parse the CAMI dataset is to unzip compressed fastq files, and then convert the fastq format into fasta format. After converting files to fasta format, a large number of reads are included in each fasta file, and each read is separated by the '>' tag and their unique header. But NBC++ wants to place each sequence into a subfolder of the source directory, by the name of its respective class. Therefore, we developed a python script named InputSplit.py for NBC++ that automatically parse those input fasta files to the form that NBC can take as input. This python script separates each read into an individual fasta file saved in the same folder and each file is named by its header.

And also, in InputSplit.py, it divides 700,000 reads into 7 folders so that we could upload 7 tasks to reduce the total running time of testing.

For the testing step, we need to run the kmr_counting.sh again to count the kmer of CAMI datasets and then run the nb-classify.bash script for each folder to test the CAMI dataset.

IV. Results

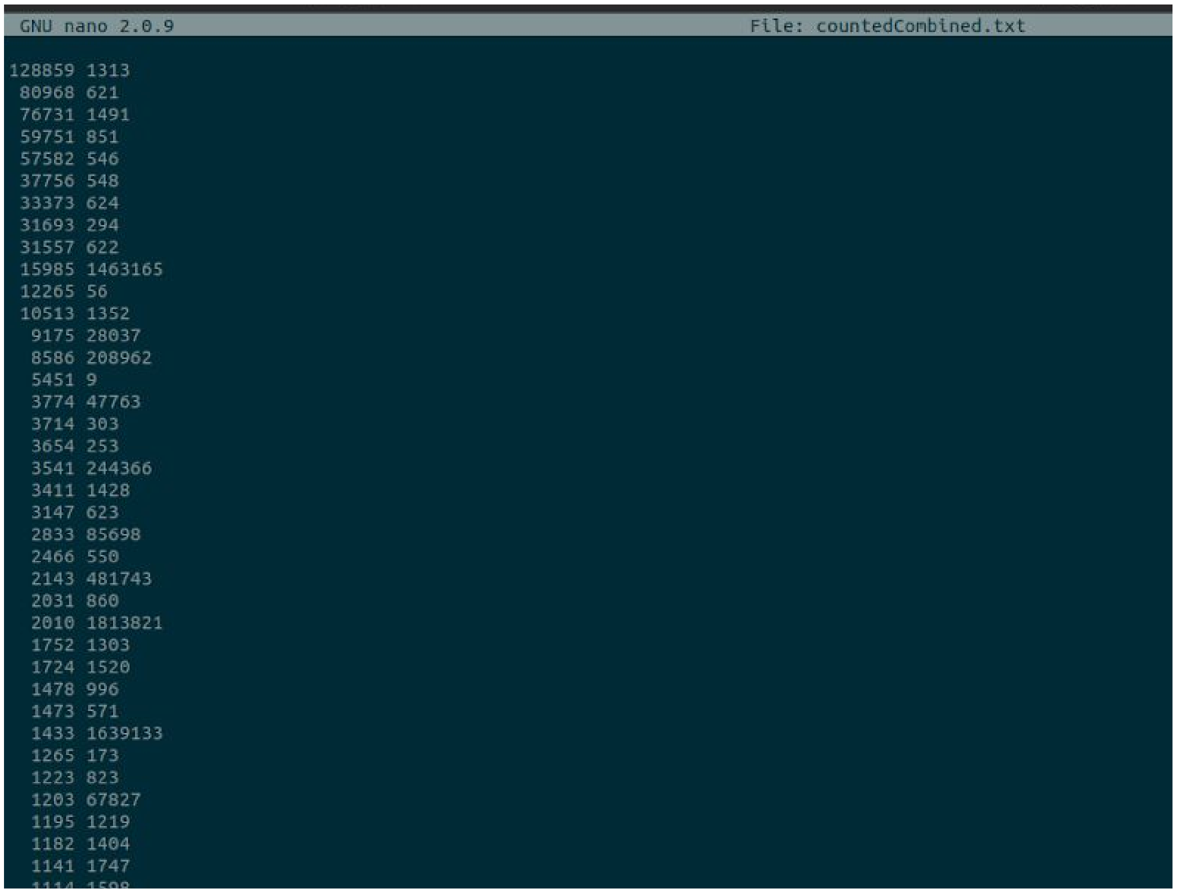
The Naive Bayes Classifier was first trained on the Genomes custom dataset provided. We trained the classifier for 15 kmer count. We performed the kmer counting using Jellyfish tool. The kmer files were generated for each of the training data files containing whole genome sequences. For the testing data we tested the classifier on the strain madness dataset provided in the CAMI Challenge. We first ran a script to extract each genome and copy it into a file. the Jellyfish tool for counting the 15 kmers in genome files. We then used the classifier to predict the TaxIDs of the short reads present in each file. The draft results after NBC prediction is shown in Figure 6. It computes the confidence for each class first, then predicts which read belongs to which class.



```
(4051/4059) Computing confidence for class 53336
(4052/4059) Computing confidence for class 2500543
(4053/4059) Computing confidence for class 29430
(4054/4059) Computing confidence for class 323415
(4055/4059) Computing confidence for class 1968541
(4056/4059) Computing confidence for class 61647
(4057/4059) Computing confidence for class 182337
(4058/4059) Computing confidence for class 1332264
(4059/4059) Computing confidence for class 2483401
Genome with class testset, predicted 1313, filename: "/lustre/scratch/yy476/Naive_Bayes/tools/strainmadness/r
Genome with class testset, predicted 546, filename: "/lustre/scratch/yy476/Naive_Bayes/tools/strainmadness/ra
Genome with class testset, predicted 851, filename: "/lustre/scratch/yy476/Naive_Bayes/tools/strainmadness/ra
Genome with class testset, predicted 622, filename: "/lustre/scratch/yy476/Naive_Bayes/tools/strainmadness/ra
Genome with class testset, predicted 1491, filename: "/lustre/scratch/yy476/Naive_Bayes/tools/strainmadness/r
Genome with class testset, predicted 1313, filename: "/lustre/scratch/yy476/Naive_Bayes/tools/strainmadness/r
Genome with class testset, predicted 9, filename: "/lustre/scratch/yy476/Naive_Bayes/tools/strainmadness/raw/
Genome with class testset, predicted 1703250, filename: "/lustre/scratch/yy476/Naive_Bayes/tools/strainmadnes
```

Figure 6. Draft Results after NBC prediction

We then ran a script to parse the output from the above file and extract the predicted TaxID for each genome. The percentage of each sequence of specific TaxID present in the sample was calculated. The output below shows the result of counting total number of each species present in the sample.

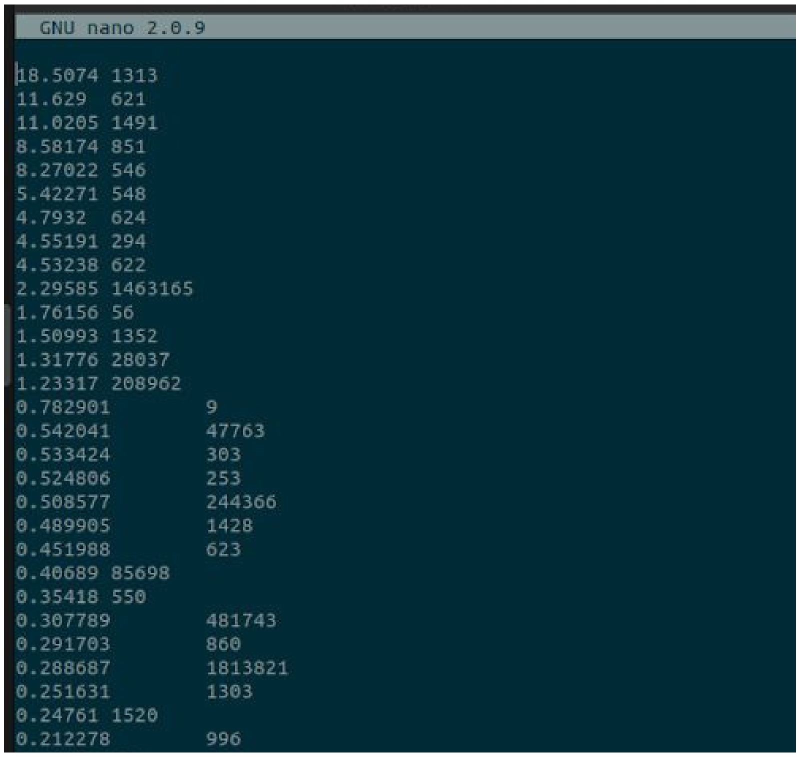


```
GNU nano 2.0.9                                              File: countedCombined.txt

128859 1313
 80968 621
 76731 1491
 59751 851
 57582 546
 37756 548
 33373 624
 31693 294
 31557 622
 15985 1463165
 12265 56
 10513 1352
  9175 28037
  8586 208962
  5451 9
  3774 47763
  3714 303
  3654 253
  3541 244366
  3411 1428
  3147 623
  2833 85698
  2466 550
  2143 481743
  2031 860
  2010 1813821
  1752 1303
  1724 1520
  1478 996
  1473 571
  1433 1639133
  1265 173
  1223 823
  1203 67827
  1195 1219
  1182 1404
  1141 1747
```

Figure 7. Result of counting the total number of sequences belonging to each species

The percentage of sample belonging to each species was then calculated and the output results can be seen in Figure 8.

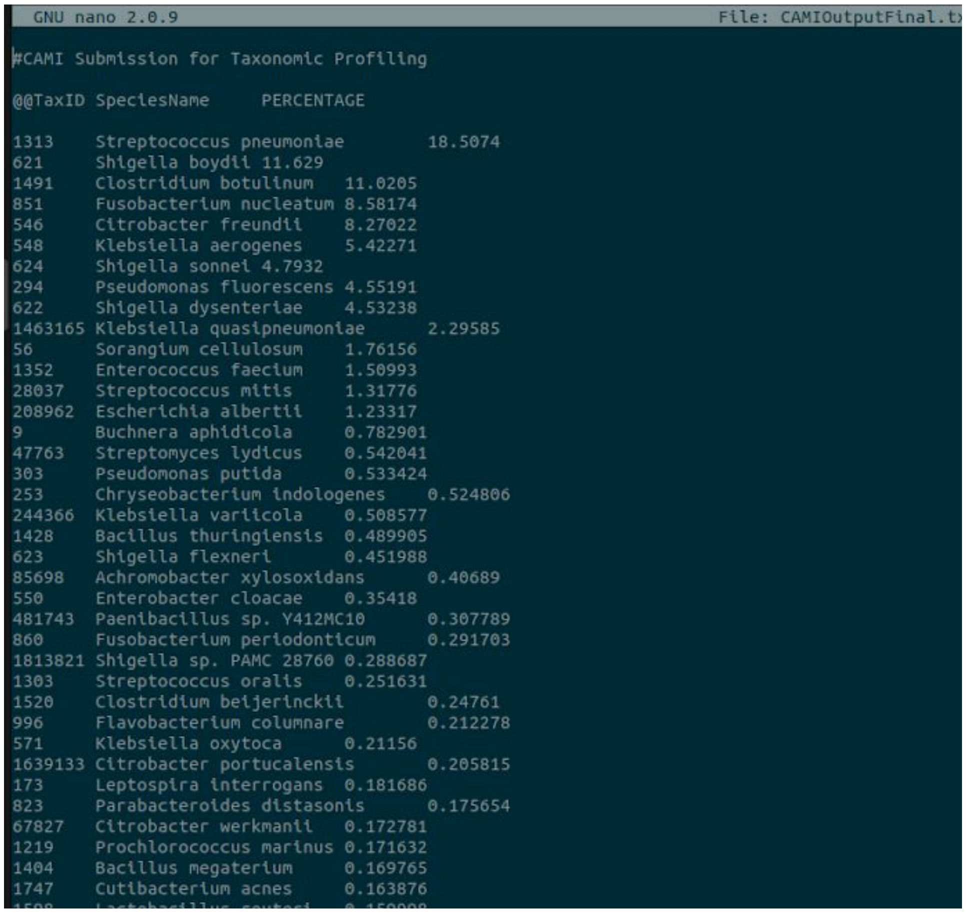


```
GNU nano 2.0.9

18.5074 1313
11.629  621
11.0205 1491
8.58174 851
8.27022 546
5.42271 548
4.7932  624
4.55191 294
4.53238 622
2.29585 1463165
1.76156 56
1.50993 1352
1.31776 28037
1.23317 208962
0.782901        9
0.542041        47763
0.533424        303
0.524806        253
0.508577        244366
0.489905        1428
0.451988        623
0.40689 85698
0.35418 550
0.307789        481743
0.291703        860
0.288687        1813821
0.251631        1303
0.24761 1520
0.212278        996
```

Figure 8. Percentage of sequences belonging to each species

We then extract the species name using the respective taxonomic identification numbers from NCBI database using a biopython script. The output result thus contains the TaxID, Species names and the percentage of species present in the sample. The final output is shown in Figure 9.



```
  GNU nano 2.0.9                                                  File: CAMIOutputFinal.tx

#CAMI Submission for Taxonomic Profiling

@@TaxID SpeciesName     PERCENTAGE

1313    Streptococcus pneumoniae        18.5074
621     Shigella boydii 11.629
1491    Clostridium botulinum   11.0205
851     Fusobacterium nucleatum 8.58174
546     Citrobacter freundii    8.27022
548     Klebsiella aerogenes    5.42271
624     Shigella sonnei 4.7932
294     Pseudomonas fluorescens 4.55191
622     Shigella dysenteriae    4.53238
1463165 Klebsiella quasipneumoniae      2.29585
56      Sorangium cellulosum    1.76156
1352    Enterococcus faecium    1.50993
28037   Streptococcus mitis     1.31776
208962  Escherichia albertii    1.23317
9       Buchnera aphidicola     0.782901
47763   Streptomyces lydicus    0.542041
303     Pseudomonas putida      0.533424
253     Chryseobacterium indologenes    0.524806
244366  Klebsiella variicola    0.508577
1428    Bacillus thuringiensis  0.489905
623     Shigella flexneri       0.451988
85698   Achromobacter xylosoxidans      0.40689
550     Enterobacter cloacae    0.35418
481743  Paenibacillus sp. Y412MC10      0.307789
860     Fusobacterium periodonticum     0.291703
1813821 Shigella sp. PAMC 28760 0.288687
1303    Streptococcus oralis    0.251631
1520    Clostridium beijerinckii        0.24761
996     Flavobacterium columnare        0.212278
571     Klebsiella oxytoca      0.21156
1639133 Citrobacter portucalensis       0.205815
173     Leptospira interrogans  0.181686
823     Parabacteroides distasonis      0.175654
67827   Citrobacter werkmanii   0.172781
1219    Prochlorococcus marinus 0.171632
1404    Bacillus megaterium     0.169765
1747    Cutibacterium acnes     0.163876
```

Figure 9. Output displaying the TaxID, species names and percentage.

From the classification results it can be seen that the Streptococcus pneumoniae has the highest percentage of 18.5074 of the sample. This species is the most abundant in the sample. Shigella boydii and Clostridium botulinum constitute around 11.629 and 11.0205 percentage of the sample. Fusobacterium nucleatum and Citrobacter freundii constitute around 8.58 and 8.27 percentage of the sample. These species constitute the majority in the sample of Strain Madness provided by the CAMI challenge.

V. Discussion

From the results it can be observed that the dataset provided by CAMI contains a relatively high percentage of Streptococcus pneumoniae 18.5074%, Shigella boydii 11.629% and Clostridium botulium 11.0205%. The Classifier also has high classification results after being trained on the Genome dataset. Shigella boydii and Clostridium botulium constitute around 11.629 and 11.0205 percentage of the sample. Fusobacterium nucleatum and Citrobacter freundil constitute around 8.58 and 8.27 percentage of the sample. These species constitute the majority in the sample of Strain Madness provided by the CAMI challenge.

Taxonomic ID	**Species Name**	**Sequence Count**	**Percentage of Sample**
1313	Streptococcus pneumoniae	128859	18.5074
621	Shigella boydii	80968	11.629
1491	Clostridium botulium	59761	11.0205
851	Fusobacterium nucleatum	37643	8.58
546	Citrobacter freundil	33456	8.27

Further analysis based on the genomic content of the species of interest can highlight the presence of specific traits of a species within a complex microbial community. For example, it would be useful for epidemiological and microbial surveillance to profile and trace directly specific antibiotic resistance genes or virulence factors.

The presence of 18% of sample containing Streptococcus pneumoniae tells us there is a high content of Streptococcus genes in the sample. Further analysis of these specific genes at a deeper level can unravel the diversity of specific microbial genetic traits among complex communities. These analysis can be included to see if there are any differences in profiles obtained from different samples, or at different time points. Thus potential pathogenic species can be recovered, typed, and traced across microbial communities that are wider and more complex than the ones we observed in the strain madness dataset. The phylogenetic relation of strains in the same species along with their functions can be simultaneously profiled which will provide a more complete characterization of strains in the samples.

Reference

[1] Peter Menzel, Kim Lee Ng, Anders Krogh: Fast and sensitive taxonomic classification for metagenomics with Kaiju. Nature Communications, 13 Apr. 2016

[2] Gail, Rosen. Drexel University EESI Lab, 2017. A Naive Bayes classifier for metagenomic data. GitHub repository, https://github.com/AlexCristian/Naive_Bayes

[3] Derrick E Wook, Steven Salzberg: Kraken ultrafast metagenomic sequence classification using exact alignment. Genome Biology 2014, 15:R46.

[4] Altschul S, Gish W, Miller W, Myers E, Lipman D: Basic local alignment search tool. J Mol Biol 1990, 215:403–410.

[5] Rosen G, Garbarine E, Caseiro D, Polikar R, Sokhansanj B: Metagenome fragment classification using N-mer frequency profiles. Adv Bioinformatics 2008, 2008:1–12.

[6] Huson D, Auch A, Qi J, Schuster S: MEGAN analysis of metagenomic data. Genome Res 2007, 17:377–386.

[7] Brady A, Salzberg SL: Phymm and PhymmBL: metagenomic phylogenetic classification with interpolated Markov models. Nat Methods 2009, 6:673–676.

[8] Brady A, Salzberg S: PhymmBL expanded: confidence scores, custom databases, parallelization and more. Nat Methods 2011, 8:367.