2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

Semi-supervised and Incremental Sequence Analysis for Taxonomic Classification

Adriana Fasino
Electrical and Computer Eng.
Rowan University, Glassboro, NJ, USA
fasino97@students.rowan.edu

Emrecan Ozdogan
Electrical and Computer Eng.
Rowan University, Glassboro, NJ, USA
ozdoga67@rowan.edu

Bahrad A. Sokhansanj
Electrical and Computer Eng.
Drexel Univ., Philadelphia, PA, USA
bahrad@molhealtheng.com
ORCID ID: 0000-0002-5050-5926

Gail Rosen
Electrical and Computer Eng.
Drexel Univ., Philadelphia, PA, USA
glr26@drexel.edu
ORCID ID: 0000-0003-1763-5750

Robi Polikar
Electrical and Computer Eng.
Rowan University, Glassboro, NJ, USA
polikar@rowan.edu
ORCID ID: 0000-0002-2739-4228

***Abstract*—Metagenomic analysis is vital in determining what organisms are present in a microbial sample and why they are present. In this study, we explore the utility of MMseqs2, a bioinformatics pipeline, for taxonomic classification in metagenomics, focusing on 16S rRNA gene sequences. We evaluate the algorithm's performance in full dataset as well as batch-by-batch incremental processing, and more importantly, we add the capability of semi-supervised classification to this otherwise clustering-only algorithm. Incremental updating is important because it allows seamless integration and processing of new data, whereas semi-supervised classification allows taxonomic identification of previously unknown organisms. We also evaluate the different clustering modes offered by MMseqs2, and compare MMseqs2 to our previously developed semi-supervised incremental algorithm SSI-VSEARCH. We show that MMseqs2's built-in *clusterupdate* function works well, and our semi-supervised classification capability adds new functionality to this bioinformatics processing pipeline.**

I. Introduction

Metagenomic samples contain genetic information from many organisms, including samples that have not previously been identified, or could not be grown in a lab. While metagenomic sequences can include all of an organism's genes, the 16S ribosomal RNA (rRNA) gene is one of the most useful biomarkers for taxonomic classifications of prokaryotes (Bacteria and Archaea). These genes consist of multiple conserved regions, which have extremely few mutations over prokaryotes, as well as variable regions, which slowly accrue mutations over the millions/billions years of evolution that can then be used to distinguish prokaryotic lineages. The 16S rRNA gene's slow mutation rate can make it difficult to distinguish between 16S rRNA genes of closely related organisms [1], [2], for example, organisms that may have diverged evolutionarily only tens of thousands of years ago (true for some species). However, 16S rRNA sequence can be used to distinguish upper-level taxonomic classes (genera and above, which have diverged hundreds of thousands/millions years ago and for phyla, up to billions of years ago), and thus it is commonly used as a marker for taxonomic classification. Databases of 16S rRNA sequences are rapidly being expanded as new organisms and microbiomes are being sequenced. These samples are collected from many different locations, including soil, water, and the human gut, and have variety of organisms living together. The genetic material directly obtained from environmental metagenomic samples – when analyzed – allows the study of entire microbial communities without the need for individual isolation or cultivation. Therefore, there is a need to predict the taxonomic labels of organisms in a metagenomic sample, as well as to identify novel (previously unknown) organisms.

Bioinformatics pipeline suites such as MG-RAST [3] [4], U/VSEARCH [5], [6], DIAMOND [7], or MMseqs2 [8], can be used to analyze and sort genetic sequences, however, these programs generally use unsupervised clustering to sort the genetic information, and then compare the results to large repositories through BLAST [9] to match to known sequences. There are two main shortcomings with these approaches: 1) they generally process the data in a single batch, expecting the entire data to be available all at once, yet in practice genomic data only become available in small batches over long periods of time; and 2) as purely clustering algorithms, they lack the ability of taxonomic classification without cross-referencing with BLAST (another widely used bioinformatics suite for searching, aligning and comparing DNA and RNA sequences against a known database of sequences [9]).

Regarding lack of capability to process the data incrementally, current approaches rely on retraining from scratch – using a combination of old and new data – when faced with new data that has recently become available. Such an approach not only requires saving and reprocessing previously processed data, it also takes extra time and resources, making no use of the clusters that were previously made. As the amount of genomic data grow, such a retraining-from-scratch approach

This work is supported by U.S. National Science Foundation under grants #1936782 & #1936791.

becomes computationally infeasible. An incremental approach, allowing the algorithm to update existing clusters without requiring access to prior data, and add new ones as needed with new batches, is much more efficient [10], [11].

Regarding the second shortcoming, as clustering-only algorithms, these approaches group sequences with similar characteristics into clusters, based on a sequence similarity threshold. As such, these algorithms provide information about organisms that share common features but do not explicitly reveal their individual identities. To be able to answer these more complex questions of "what organisms are there?," a supervised classification algorithm is needed. Of course, any supervised training algorithm requires previously curated training data with known ground truth labels. In real world metagenomics, curated datasets with correct label information, i.e., *reference datasets*, are much smaller in size and less readily available when compared to widely available, unlabeled experimental and environmental datasets. Taking advantage of any available labeled training data and combining them with much larger unlabeled datasets matches the machine learning communities' definition of *semi-supervised learning*, which allows us to strategically guide an otherwise unsupervised clustering algorithm in predicting actual labels.

We have previously developed incremental and semi-supervised versions of VSEARCH (I-VSEARCH and SSI-VSEARCH) , adding incremental processing and classification capabilities to VSEARCH, another common bioinformatics pipeline used to cluster 16SrRNA genes [11], [12]. More recently, a newer bioinformatics pipeline platform, called MMseqs2, became available that has a built incremental processing capability for searching and clustering [8], although it does not have a supervised classification capability. Due to its built-in incremental update function, we now focus on this algorithm: we evaluate its incremental learning module, add a new semi-supervised learning capability, and compare it to SSI-VSEARCH. We report our results on the same version of the Ribosomal Database Project (RDP) full-length 16s rRNA gene dataset [13] that we used in exploring the proprieties of SSI-VSEARCH [12].

II. Background

A. Genetic Clustering

Several clustering methods have been developed to address the specific needs of metagenomics. A popular approach is CD-HIT [14], known for its efficiency, which employs a sequence similarity threshold to create clusters: if a sequence is sufficiently similar to the cluster representative (also known as *seed* or *centroid*) of an existing cluster, it is placed in that cluster, otherwise a new cluster is created with the new sequence serving as its seed. All other approaches use some variation of CD-HIT's clustering framework, and are generally greedy, i.e., each new sequence is clustered with the first cluster representative that matches at or above the similarity threshold. VSEARCH, the open-source version of USEARCH, adds a slight modification: it aligns sequences globally to determine the similarity, which minimizes memory usage during clustering. MMseqs2, one of the newest approaches, uses a cascaded-clustering process in its default mode in order to refine the clustering results. Another pipeline is BLASTclust, a program provided by NCBI's BLAST software package. BLASTclust also uses a greedy policy with single-linkage approach [15], which makes it one of the more accurate, but also the slowest, choices. Overall, choosing the most suitable clustering method depends on factors such as the size of the dataset, available computational resources, and the desired level of sensitivity.

B. MMseqs2

MMseqs2 (many-against-many sequence searching 2) is a bioinformatics software suite known for its rapid and efficient sequence analysis and comparison capabilities [8] [16]. MMseqs2 has a similar sensitivity to BLAST, but 400 times faster. MMseqs2 is the successor to the original MMseqs algorithm [17] and is specifically designed to address the need for high-speed and accurate sequence searching in large-scale datasets. An important distinction of MMseqs2 is that it has different clustering modes, including a built-in incremental clustering option, described in more detail below. We note that all clustering modes of MMSeqs2 are unsupervised clustering algorithm and do not use any labels, even if they are available. We therefore add a simple and efficient majority-vote based semi-supervised approach to MMseqs2 to provide it with the taxonomic classification capability.

III. Methods

A. Cluster Modes and Options

MMseqs2 uses the basic clustering mechanism of comparing each query sequence to each cluster representative (seeds, centroids), computes a similarity measure, referred to as sensitivity, and compares it to a preset threshold. Sensitivity is the percentage of the k-mers (k-long sequence fragments) that need to match between the query and the seed. When MMseqs2 finds a sequences for which the similarity meets or exceeds the threshold, the query sequence is placed into that cluster. If no such cluster is found, a new cluster is formed with the query sequence serving as its seed. MMseqs2 offers three cluster modes, each slightly modifying the above-stated basic mechanism to determine how the algorithm forms the clusters. The default cluster mode is the *Greedy Set* cover (also known as cluster mode 0), which works by selecting the sequences with the most sensitivity matches and creating clusters around those nodes. The seeds (centroids) of these clusters are chosen based on the sequences that have the most similarity matches. Once all clusters are formed, a reassignment step compares each sequence to each centroid via an alignment score. Clusters are reformed if any of the outlying sequences do not meet the predetermined similarity threshold. The next option is *connected component*, also known as cluster mode 1, where clusters are formed by choosing a sequence that has the most similarity matches in a certain group and making it the centroid. The cluster is then formed around that centroid. This approach results in fewer clusters that are both large and

less specific. The main difference between cluster mode 0 and 1 is that the former breaks up a larger group if it can find another sequence with enough similar sequences to become its own centroid, whereas the latter does not. The final option is *greedy incremental*, cluster mode 2, which is similar to the original CD-HIT clustering algorithm, as both algorithms chose their centroids based on the length of the sequence. This approach takes the next longest sequence, makes it the centroid and forms a cluster using the sequences that pass the similarity threshold, and then moves on to the next longest sequence in the dataset. These three clustering modes are illustrated in Figure 1. We evaluate and compare these three modes later in the Results section.

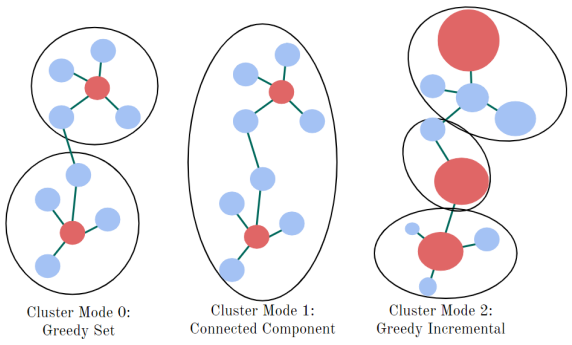

Fig. 1. Three clustering modes offered by MMseqs2: greedy set, connected component, and greedy incremental. Here, cluster centroids are red, other sequences are blue nodes. A line represents a successful similarity match between nodes. Greedy set chooses centroids based on the nodes that has the most connections. Connected component also finds the nodes with the most connections, but will only look for one per group where any connections are found and will group all nodes together. Greedy incremental clusters based on the length of the genetic sequences. In this example, the size of the node represents the length of the genetic sequence, so the larger nodes are the centroids with the longest sequence, and are clustered with smaller nodes that meet the sensitivity threshold.

B. Clusterupdate and Evaluation

Of main interest is MMSeq2's *clusterupdate* option, which allows MMseqs2 to process new batches of datasets in an incremental manner. This function retains existing clusters, giving preference to sorting new query sequences into existing clusters before forming new ones. Typical usage is to use one of the three modes for the initial clustering, then using the *clusterupdate* function to add new data.

The algorithm's performance and behavior are assessed using a variety of metrics. To truly evaluate its predictive capability, we relied on the ground truth. For this purpose, we used a 16S rRNA dataset from Ribosomal Database Project (RDP) [13], which provided us with the necessary label information. In order to mimic scenarios with known and unknown labels, we partitioned the entire dataset into two subsets: a training dataset and a test set. To simulate a relatively small labeled data and a larger unlabeled data, we used %25 of the total data as training (reference) dataset, while the remaining %75 represented experimental/testing (unlabeled) data. During the clustering stage, the algorithm did not use any of the labels. Recall that MMseqs2 is entirely unsupervised and any labels that may be available are not used. During our postprocessing, we attached any available training data labels to their sequences, based on which cluster labels are obtained through majority voting (as described below). We deliberately concealed the labels of the *test data* during post-processing when completing the majority vote to simulate unlabeled data and to ensure unbiased evaluation. We then utilized these hidden labels of the test data as the ground truth to calculate the performance metrics as described below.

1) Semi-supervised Learning Through Majority Voting: The core clustering process for all genomic pipeline algorithms is purely an unsupervised procedure. We add a semi-supervised learning (SSL) functionality to this pipeline through a simple majority voting algorithm. Once the clusters are formed (during which no labeled information is used), we check which sequences in any given cluster actually come from the labeled training data. Given that the dataset consists of both labeled (few) and unlabeled (many) sequences, each cluster may include a set of labeled or unlabeled sequences.

If there is only one labeled sequence in a cluster, that label becomes the label of that cluster and that of all sequences in that cluster. If there are multiple labeled sequences in the cluster, we compute the majority vote of the known labels, and the label with the most associated sequences in that cluster becomes the cluster label, and the label of all of its *unlabeled* (test) sequences. Previously known labels of the training data in that cluster are *not* changed, even if the cluster gets a different label based on majority voting. Rare ties are broken randomly. All such sequences labeled by this approach are then considered as *predicted sequences* and their overall accuracy is computed as *predicted accuracy*. If a cluster is populated with only unlabeled sequences, we refer to it as an *unlabeled cluster*, and assign a temporary label. Unlabeled clusters are given their own accuracy metric, *unlabeled accuracy*, as defined below. The sequences in an unlabeled cluster are considered *novel* sequences. If a reference sequence with a known label is *later* added to a previously unlabeled cluster in the future (through the incremental processing), the temporary label is replaced with new the true label. This approach provides a mechanism for any truly novel organism to be identified and properly labeled when that organism is sequenced, given a name, and later appear in a future sample.

2) Prediction Accuracy: Prediction accuracy is the accuracy applied to labeled clusters. This metric is only computed on the test dataset within labeled clusters, so that it is not artificially inflated due to known labels of the training data. Majority voting using training data is employed to predict the overall label for the cluster, and any unlabeled sequences are given this predicted label. This predicted label is then compared to the ground truth, and the ratio of correct predictions is computed as the prediction accuracy: $PA = PS_{correct}/(PS_{correct}+PS_{incorrect})$ where, $PS_{correct}$ and $PS_{incorrect}$ are the number of predicted sequences in the test dataset whose labels are determined to be correct or incorrect, respectively, compared to ground truth.

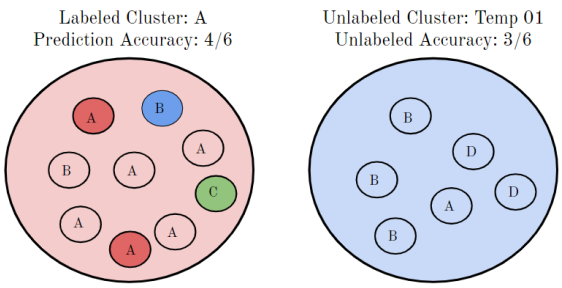

Fig. 2. Labeled (left) and unlabeled (right) clusters. Each small circle represents a genetic sequence with a different label. Solid filled circles represent labeled (reference) sequences, and the unfilled circles are unlabeled (experimental/test) data (known ground truth indicated in the circles are hidden from the algorithm). After majority voting, the left cluster is labeled as A. When computing predicted accuracy, we only consider samples from unlabeled test dataset, so the predicted accuracy is 4/6. The unlabeled cluster receives a temporary label, but the ground truth labels are used in post-processing to calculate the unlabeled accuracy. The majority vote of the ground truth labels gives the cluster and overall label of B, so the unlabeled accuracy is 3/6, since three of the sequences in this cluster are of label "B".

3) Unlabeled Accuracy: Recall that unlabeled clusters contain no reference data, and therefore are assigned a temporary label by our SSL algorithm. In a real world scenario, the experimental test sequences may come from truly unknown novel organsims. Our incremental SSL algorithm allows such scenarios, and leaves the cluster with its temporary label until such time a sequence with a known label is placed into that cluster - perhaps at some time in the future when that novel organism is sequenced, given a name, and then appears in a future reference dataset. For the purpose of this experiment, we do have the ground truth labels for all data, but we hide them from the algorithm. Therefore, we can actually compute the accuracy of these "unlabeled" testing sequences in being placed into clusters during post-processing. We do so by reassigning the ground truth labels to the test sequences during evaluation, and invoking the majority vote to assign the cluster with a label. This label is then compared to the ground truth labels and an accuracy metric is calculated. We refer to this metric as *unlabeled accuracy*, which can be calculated as $UA = US_{correct}/(US_{correct}+US_{incorrect})$ where, $US_{correct}$ and $US_{incorrect}$ are the number of (unlabeled) test data sequences in an unlabeled cluster whose labels are determined to be correct or incorrect, respectively, compared to ground truth data. Unlabeled accuracy is an important metric, because it captures and measures the underlying ability of the algorithm to form correct clusters.

4) Singletons: Singletons are sequences that are not similar enough to any other sequence to be placed into a cluster. Singletons, labeled or not, are not included in our accuracy calculations to ensure that our accuracy results are not artificially increased. However, we do keep track of – and report – the number of singletons.

IV. Experimental Setup

A. RDP18 Dataset

We used the RDP18 dataset, the 18^{th} (2020) release of the 16S rRNA dataset provided by the Ribosomal Database Project [13]. The dataset consists of 21,195 16S rRNA sequences from Bacteria and Archaea. 20,198 of these have six levels of taxonomic rank information (Kingdom, Phylum, Class, Order, Family and Genus), so we further curated the dataset to include just these sequences for our testing purposes. We also removed 6 phyla that had fewer than four total samples. The final curated dataset had 20,186 samples from 32 phyla. We partitioned this dataset into two subsets, with 25% to be used as labeled (reference/training) data and the remaining 75% to be used as unlabeled (experimental/test) data. The training data was then further divided into five equal batches to simulate incremental learning.

B. Experiments

We compared the ability of the MMseqs2 algorithm - with the added semi-supervised learning capability - to learn incrementally and provide taxonomic classification for novel sequences. We used all three clustering modes (cluster mode 0 - greedy set, cluster mode 1 - connected component, and cluster mode 2 - greedy incremental) run with full data in a single batch as well as incrementally with five batches. The *percent identity similarity threshold* is the primary free-parameter of the algorithm, and determines how similar the genetic sequences must be to be clustered together when compared to a cluster centroid. We performed a parameter sweep, by testing a range of eight similarity threshold values in the commonly used interval of 0.75 to 0.97 (i.e., 0.75, 0.78, 0.81, 0.84, 0.87, 0.90, 0.93, 0.97) for both the full and incremental runs. Each one of those 16 runs was performed 10 times in order to prove repeatability and accuracy.

Clustering algorithms, due to their nature, are usually very sensitive to the order in which the data are presented. To test the robustness of our approach against the order of data presentation, we have also added another experiment, where the entire dataset was shuffled before splitting into labeled (training) / unlabelled (test) datasets, as well as partitioning into five batches.

V. Results and Discussion

A. Default Cluster Mode 0 - Full vs. Incremental

Figures 3 and 4 show predicted and unlabeled accuracy, respectively, when MMseqs2 is run in cluster mode 0, both using the entire dataset in a single run (full) and using each of the batches one at a time (incremental). We note that neither predicted nor unlabeled accuracy count singletons, as majority voting on clusters of a single sequence is meaningless, and counting them would unfairly inflate the performances.

In Figs. 3 and 4, the solid and dashed lines represent the full and incremental runs, respectively. For incremental runs, the results are those obtained after the processing of the 5^{th} batch. The different colors represent the different levels of

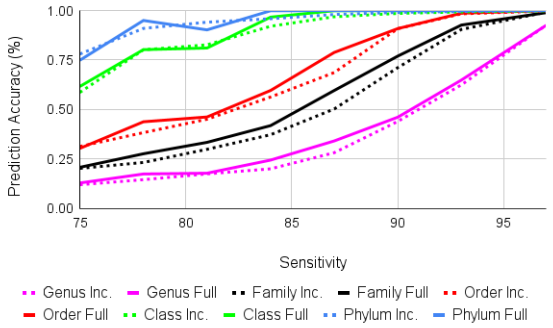

Fig. 3. Cluster Mode 0 Prediction Accuracy Incremental vs. Full

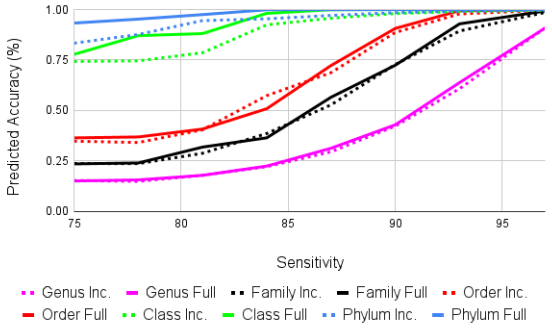

Fig. 5. Randomized Cluster Mode 0 Prediction Accuracy Incremental vs. Full

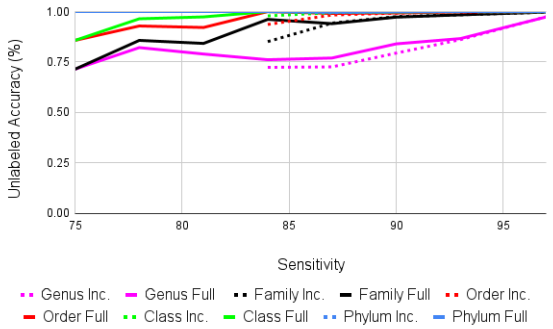

Fig. 4. Cluster Mode 0 Unlabeled Accuracy Incremental vs. Full

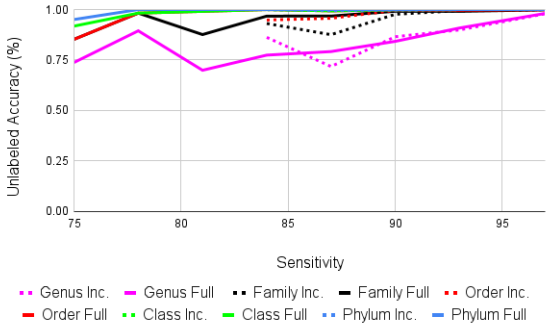

Fig. 6. Random Cluster Mode 0 Unlabeled Accuracy Incremental vs. Full

taxonomic depth – from the most specific (genus) to most general (phylum). The main observation here is that at each taxonomic level, both the full and incremental runs perform similarly, and follow the same trends. This is critical, because it demonstrates that we can achieve the same accuracy even when we have to process only a faction of the data at a time, and when we do not have the luxury of having the entire dataset at once for processing. We also observe that the algorithm's SSL module work as intended, providing taxonomic classification whose accuracy improves – as expected and desired – with sensitivity and with taxonomic level. Note that when sensitivity is increased, the sequences need to be more similar to be placed in the same cluster. Of course, the cost of higher accuracy at higher sensitivity thresholds is the longer run times. We also note in Fig 4 that we do not have unlabelled accuracy results for low sensitivity levels in incremental runs; this is because at low sensitivity the clusters are large enough that there is at least one labeled sequence in each cluster, so only a predicted accuracy can be computed.

As mentioned earlier, clustering algorithms are sensitive to the order in which the data are presented. To determine the robustness and repeatability of our semi-supervised MMseqs2 to order of data presentation, we repeated the above experiment, but with a completely and randomly shuffled dataset. Figures 5 and 6 show the results of the randomized experiment.

We observe that the results and the trends remain the same, demonstrating that the incremental use of MMseqs2 with the semi-supervised classification performs very well, and similarly to running the algorithm on the full dataset at once. Since real world data can only be obtained incrementally in batches over a period of time, these results show that MMseqs2 with semi-supervised classification is now a viable option for real world processing of metagenomic sequences.

B. Comparing Cluster Modes

In its default mode (cluster mode 0), MMseqs2 performed very well, but we also wanted to evaluate the other clustering modes to determine whether they offer any advantages. Clustering mode 1 is also of interest to us, as it is similar to the clustering approach used by VSEARCH, providing us with a more fair comparison against SSI-VSEARCH. Figures 7 and 8 illustrate the predicted and unlabeled accuracy of semi-supervised MMseqs2 when used in cluster mode 1 (connected component) in full and incremental settings. As in prior experiments, we show the results at all taxonomical levels and similarity thresholds.

Cluster mode 1 produces fewer and larger clusters compared to cluster mode 0, resulting in faster run times, but slightly lower accuracy of sequence matching. The results reflect this expectation primarily at the less specific taxonomic depths such as phylum and class and generally at lower sensitivity thresholds (where we would normally expect less accuracy). A pleasant surprise here was that the incremental runs performed better than full (single batch) runs, producing higher predicted

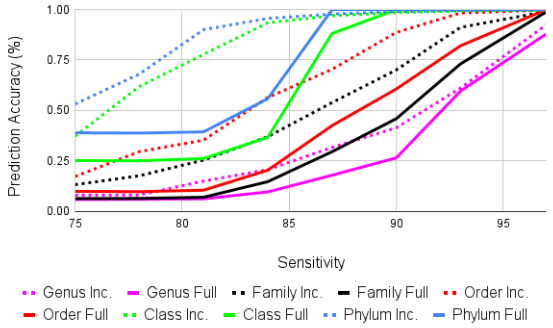

Fig. 7. Cluster Mode 1 Predicted Accuracy Incremental vs. Full

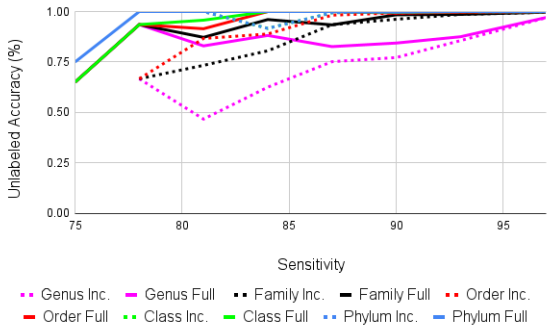

Fig. 8. Cluster Mode 1 Unlabeled Accuracy Incremental vs. Full

accuracy throughout every sensitivity and taxonomic level. The results were more mixed for the unlabeled accuracy, but this metric tends to have a higher variance as there are usually a lower number of clusters - particularly with any unlabeled sequences - when the clusters themselves are larger. Regardless of the specific accuracy numbers, however, the trends observed with cluster mode 0 were generally observed with cluster mode 1 as well.

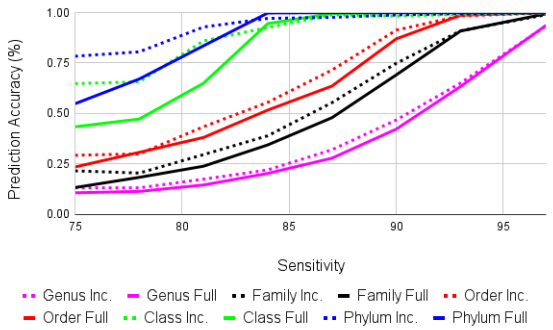

Fig. 9. Cluster Mode 2 Predicted Accuracy Incremental vs. Full

We have then evaluated cluster mode 2, whose results are shown in Figures 9 and 10. These results also followed similar trends as cluster mode 1, where our incremental semi-supervised approach performed generally better than the

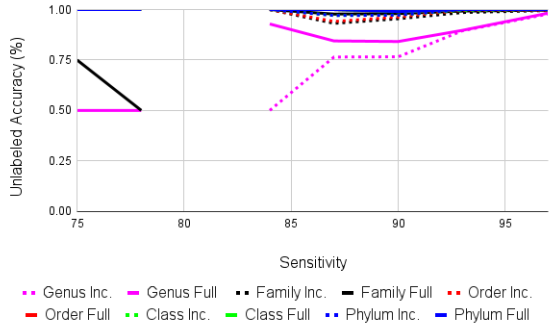

Fig. 10. Cluster Mode 2 Unlabeled Accuracy Incremental vs. Full

regular full data runs.

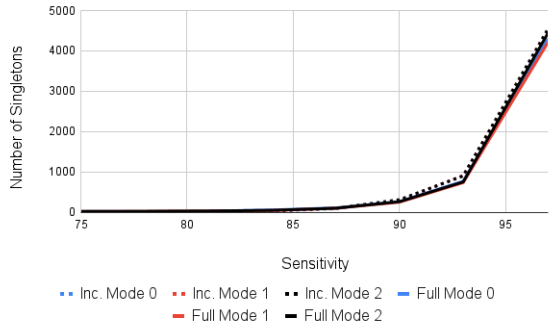

Fig. 11. Number of Singletons

Finally, we also looked at the number of singletons - clusters that consist of a single sequence. As shown in Figure 11, the number of singletons increase with sensitivity, an expected outcome. More importantly, however, the number of such singletons is virtually the same whether the algorithm is run in full with the entire dataset, or incrementally in batches. In general, then, despite some minor differences in absolute numbers, all clustering modes generally show similar trends, with cluster mode 0 providing slightly better accuracies at all taxonomic levels.

C. MMseqs2 vs. SSI-VSEARCH

Since MMseqs2 is a newer algorithm with a built-in cluster update feature (but without a semi-supervised component), we wanted to compare its results to those of our previously introduced SSI-VSEARCH [12]. SSI-VSEARCH is a semi-supervised and incremental version of the original VSEARCH algorithm, which itself is a purely clustering algorithm that lacks the ability to incrementally process the data.

Figures 12 and 13 show the prediction and the unlabeled accuracy for SSI-VSEARCH for the incremental runs (additional results can be obtained in [12]).

When comparing these results to MMseqs2 results shown in the previous sections, we observe that the results and the trends are indeed very similar, but have a few main distinctions.

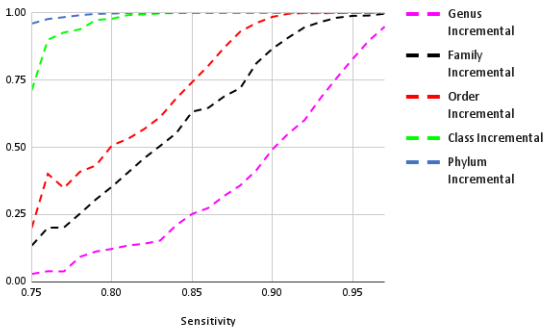

Fig. 12. Predicted Accuracy for Incremental Runs Processed by SSI-VSEARCH

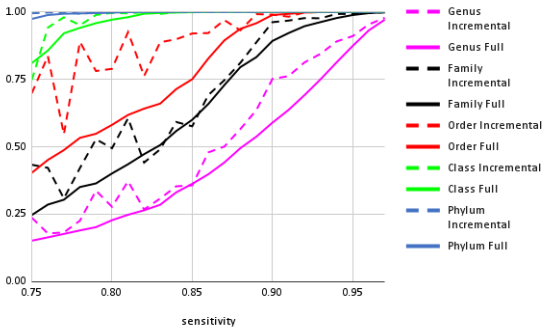

Fig. 13. Unlabeled Accuracy for Full and Incremental Runs Processed by SSI-VSEARCH

SSI-VSEARCH tends to perform better at less specific taxonomic depths, i.e. phylum and class, but the incremental runs have higher variability. MMseqs2, across all cluster modes, follows smoother patterns, but overall the results and trends are (pleasantly) surprisingly similar. The main takeaway from comparing these results is that both approaches perform well, particularly in an incremental semi-supervised environment.

VI. Conclusions

In this study, we explored the use of MMseqs2, a bioinformatics pipeline suite for the taxonomic classification of 16S rRNA genes. We evaluated its ability to learn incrementally, and also with the added semi-supervised learning capability to provide taxonomic classification – a capability the original MMseqs2 lacks. Using 16S rRNA genetic sequences, our modified MMseqs2 showed that it can provide high accuracy in taxonomic classification; its incremental processing performs as well as - and sometimes even better than – the full single batch processing. Both the incremental learning ability and the semi-supervised classification ability are important capabilities for processing metagenomic sequences: the incremental processing ability allows processing the data in batches, a scenario that reflects real world data collection constraints. The semi-supervised processing allows taxonomic classification, answering the all-important question ”what organisms live here?”

Our future work will continue to focus on finding the best and most useful clustering algorithms for sorting and classifying metagenomic data. There is still more to be done with MMseqs2, as there are other metrics that we would like to use to compare clustering methods. These metrics include, but are not limited to, completeness, homogeneity, v-measure, adjusted rand index, and timing statistics. Another important task is to evaluate the approach on larger datasets to see the scalability of our findings. Additional tasks we have for future work include using datasets that are more realistic, meaning they are less complete or use other variable regions, or changing the distribution of training and testing data to see how these variables can influence the results.

References

[1] Y. Lan, G. L. Rosen, and R. Hershberg. “Marker genes that are less conserved in their sequences are useful for predicting genome-wide similarity levels between closely related prokaryotic strains,” Microbiome, vol. 4, no. 1, pp. 1-18, 2016.

[2] Y. Lan, J. C. Morrison, R. Hershberg, and G. L Rosen. “POGO-DB—a database of pairwise-comparisons of genomes and conserved orthologous genes,” Nucleic Acids Research, Vol. 42, No. D1, pp. D625–D6322014, 2014.

[3] F. Meyer, D. Paarmann, M. D’Souza, et al. “The metagenomics RAST server–a public resource for the automatic phylogenetic and functional analysis of metagenomes,” BMC Bioinformatics, vol. 9, no. 1, pp. 1–8, 2008.

[4] K. P. Keegan, E. M. Glass, and F. Meyer, “MG-RAST, a metagenomics service for analysis of microbial community structure and function,” in Microbial Environmental Genomics, 1st Ed. New York City, NY, USA: Springer, 2016, pp. 207–233.

[5] R. C. Edgar, “Search and clustering orders of magnitude faster than BLAST,” Bioinformatics, vol. 26, no. 19, pp. 2460–2461, 2010.

[6] T. Rognes, T. Flouri, B. Nichols, et al. “VSEARCH: a versatile open source tool for metagenomics,” PeerJ, vol. 4, pp. e2584, 2016.

[7] B. Buchfink, C. Xie, D. H. Huson, “Fast and sensitive protein alignment using DIAMOND,” Nature Methods, vol. 12, no. 1, pp. 59–60, 2015.

[8] M. Steinegger and Söding, Johannes, ”MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets,” Nature Biotechnology, vol. 35, no. 11; pp. 1026-1028, 2017.

[9] S. F. Altschul, W. Gish, W. Miller, et al. “Basic local alignment search tool,” Journal of Molecular Biology, vol. 215, no. 3, pp. 403–410, 1990.

[10] Z. Zhao, A. Cristian, and G. Rosen, “Keeping up with the genomes: efficient learning of our increasing knowledge of the tree of life, ”BMC Bioinformatics, vol. 21, no. 1, pp. 1–23, 2020.

[11] E. Ozdogan, N. C. Sabin, T. Gracie, et al. “Incremental and Semi-Supervised Learning of 16S-rRNA Genes For Taxonomic Classification,” in IEEE Symposium Series on Computational Intelligence, 2021, pp. 1–7.

[12] E. Ozdogan, A. Fasino, R. Nguyen, et al. ”Semi-supervised and Incremnetal VSEARCH for Metagenomic Classification” in IEEE Symposium Series on Computational Intelligence, 2022, pp 1-8.

[13] J. R. Cole, Q. Wang, J. A. Fish, et al. “Ribosomal Database Project: data and tools for high throughput rRNA analysis,” Nucleic Acids Research, vol. 42, no. D1, pp. D633–D642, 2014.

[14] W. Li and A. Godzik. 2006. “Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences,” Bioinformatics, vol. 22, no. 13, pp. 1658–1659, 2006.

[15] National Center for Biotechnology Information (NCBI) Documentation of the BLASTCLUST-algorithm. Available online at ftp://ftp.ncbi.nih.gov/blast/documents/blastclust.html. Last accessed 15 August 2023.

[16] M. Steinegger and Söding, Johannes, ”Clustering huge protein sequence sets in linear time,” Nature Communications, vol. 9, no. 1, pp. 2542, 2018.

[17] M. Hauser, M. Steinegger, J. Soding, ”MMseqs software suite for fast and deep clustering and searching of large protein sequences sets, ” Bioinformatics, vol. 32, no. 9, pp. 1323-1330, 2016.