Semi-supervised and Incremental VSEARCH for Metagenomic Classification

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

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

Rachel Nguyen
Electrical and Computer Eng.
Drexel University
Philadelphia, PA, USA
rtn280@gmail.com

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

Gail Rosen
Electrical and Computer Eng.
Drexel University
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

2022 IEEE Symposium Series on Computational Intelligence (SSCI) | 978-1-6654-8768-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/SSCI51031.2022.10022184

***Abstract*—DNA Sequencing of microbial communities from environmental samples generates large volumes of data, which can be analyzed using various bioinformatics pipelines. Unsupervised clustering algorithms are usually an early and critical step in an analysis pipeline, since much of such data are unlabeled, unstructured, or novel. However, curated reference databases that provide taxonomic label information are also increasing and growing, which can help in the *classification* of sequences, and not just *clustering*. In this contribution, we report on our progress in developing a semi-supervised approach for genomic clustering algorithms, such as U/VSEARCH. The primary contribution of this approach is the ability to recognize previously seen or unseen novel sequences using an incremental approach: for sequences whose examples were previously seen by the algorithm, the algorithm can predict a correct label. For previously unseen novel sequences, the algorithm assigns a temporary label and then updates that label with a permanent one if/when such a label is established in a future reference database. The incremental learning aspect of the proposed approach provides the additional benefit and capability to process the data continuously as new datasets become available. This functionality is notable as most sequence data processing platforms are static in nature, designed to run on a single batch of data, whose only other remedy to process additional data is to combine the new and old data and rerun the entire analysis. We report our promising preliminary results on an extended 16S rRNA database.**

***Index Terms*—taxonomic classification, incremental learning, VSEARCH, semi-supervised learning**

I. Introduction

The biological diversity of the microbiome (the community of bacteria that inhabits an environmental or clinical sample) can be best observed from its genetic information. Morphological differences between cells often fail to distinguish between even distantly related microorganisms. The development of next-generation sequencing technology enabled collection of "metagenome" sequences, which contain the genetic information from potentially all the organisms in a sample, including bacteria that could not be cultured and grown in a lab. While metagenomic sequences can include all of an organism's genes, the 16S ribosomal RNA (rRNA) gene remains a useful biomarker for taxonomic classifications of bacteria, as these genes are both conserved across all bacteria and contain variable regions that can be used to construct phylogenetic trees that accurately reflect evolution [15]. With so much high-throughput sequencing capacity now available, 16S rRNA and metagenomic data have explosively grown in the last several years. This growth makes the process of identifying taxa in samples from their genetic content particularly challenging, especially when novel (previously unknown) sequences that cannot be classified are observed.

Metagenomic sequences are analyzed by online platforms, such as MG-RAST [10], or locally by U/VSEARCH [6], [13], DIAMOND [3], or MMseqs2 [11], which use unsupervised clustering approaches. Unsupervised algorithms are reasonable in this setting, since the data is often unstructured with many of the sequences not having known labels. In this paper, we focus on taxonomic classification at various depths (ranks), i.e., the phylum, family, genus, etc., that identifies what organisms are observed in a sample (answering the question "who is there?" in a given environmental sample). Notably, the methods we describe can potentially be extended to functional labels, i.e., what proteins are encoded by genetic information in a sample (answering the question "what can they do?"). In either case, however, clustering algorithms – in contrast to classification algorithms that provide a label – provide only limited information by only placing "similar" organisms, as judged by the algorithm's similarity metric, into the same cluster. As a result, clustering algorithms only tell us which organisms are alike, not who they are. Taxonomic identification requires a *supervised classification algorithm*, but these algorithms need

This work is supported by U.S. National Science Foundation under Grant #1936782.

previously curated training data with known correct labels to learn how to classify any new data. Supervised algorithms generally require substantial training datasets to create accurate classification models. But in real world biological applications, curated datasets with correct label information, i.e., reference datasets, are much smaller in size compared to experimental (and hence unlabeled) datasets. Combining (small) labeled and (large) unlabeled datasets for learning has long been known to machine learning communities as *semi-supervised learning* (SSL). Use of SSL in bioinformatics applications has been very limited [4], and is the primary focus in this effort.

Another concern with most clustering algorithms as used in genomic data analysis is that they have been static approaches, designed for a single run on a single batch of data. These algorithms, when presented with additional data that may later become available, are forced to discard all the clusters learned from previous run, combine the old data with the new data, and run the entire clustering algorithm again from scratch on the combined data. Such an approach is not only inefficient, it is increasingly untenable, intractable and simply unimplementable with the explosive growth in new datasets. A better approach would be to use incremental (continual) learning algorithms that are designed to process batches of streaming data as they become available without requiring access to – or reprocessing of – the previously analyzed data. Just like SSL approaches, incremental learning algorithms have also long been known to machine learning communities, but are only beginning to be explored for taxonomic classification applications [16].

We recently proposed Incremental VSEARCH (I-VSEARCH) as a wrapper approach to the well-established USEARCH (and its open source version VSEARCH) [12]. In this paper, we further develop and explore the unique behaviors of this algorithm through several newly defined metrics, focusing on its semi-supervised classification functionality, and its ability to recognize both previously seen or unseen novel sequences. We show that, for sequences whose samples were previously seen by the algorithm, the algorithm can predict a correct label. Perhaps more beneficial is the ability of the algorithm to process previously unseen novel sequences, assigning them a temporary label in a cluster, and then updating that label with a permanent one if/when such a label is established in a future reference database. We report our results on a new and expanded version of the Ribosomal Database Project (RDP) 16s rRNA dataset [5].

II. Background

A. Genetic Clustering

USEARCH and CD-HIT, popular clustering algorithms, follow a similar approach to genetic clustering: they first sort the sequences from longest to shortest, and compare each query sequence to the representatives (referred to as *seeds* in USEARCH) of existing clusters, where the longest sequence of each cluster serves as its representative. Both algorithms follow a greedy approach: if the similarity of sequence and a cluster representative is above a certain threshold, the search stops and the query sequence is placed into that cluster. If no matches are found, a new cluster is created and the query sequence becomes its representative. VSEARCH, which we discuss in more detail below, is essentially an open source version of USEARCH, but also differs from USEARCH during its decision making step. While USEARCH uses seed-and-extend approach, VSEARCH aligns sequences globally to determine similarity between query and target sequences. Other similar clustering algorithms include kClust [7], BLASTclust, [1], and MMseqs2, one of the latest additions to the online processing platforms and one that provides a cluster update module [11].

B. VSEARCH

Most clustering algorithms have a similar workflow. As a result, much of our proposed approach to add incremental and SSL capability is applicable to most, if not all, clustering algorithms mentioned above. However, we started our analysis by selecting VSEARCH as the base algorithm, in part because it is computationally efficient, it is open source and is well-suited for 16S rRNA databases [14]. VSEARCH consists of two steps, alignment and clustering. As mentioned above, all sequences are sorted based on their length from longest to shortest as a preprocessing step, followed by Needleman-Wunsch algorithm to align the sequences. Needleman-Wunsch is a global alignment algorithm that equalizes lengths of the sequences by adding gaps between nucleotides/proteins while pairing similar sections of different sequences.

Once the alignment is completed, VSEARCH starts clustering with a list of sorted sequences and a user-selected percent-similarity threshold, typically between 75% and 99%. VSEARCH compares each new query sequence to the seeds of each of the previously generated clusters. If the similarity between the query and any of the cluster seeds exceeds the user-selected similarity threshold, the query sequence is added to that cluster. At algorithm initiation, the longest sequence in the dataset becomes the seed of the first cluster. Then, the second sequence is compared to the seed of the first cluster, and placed into that cluster if their similarity exceeds the threshold. Otherwise, a new cluster is created with this second sequence becoming its seed. The algorithm than proceeds iteratively, with each new sequence being compared to all existing cluster seeds in order of seed length. The query sequence is placed into the first cluster for which the similarity exceeds the threshold. If none of the existing cluster seeds have a similarity higher than the threshold, then a new cluster is created with the query sequence becoming its seed. VSEARCH follows this greedy policy in placing the sequences into cluster, as each new sequence is placed into the first cluster for which the similarity threshold is met, saving considerable computation time at little or negligible cost of clustering performance.

It is important to emphasize that VSEARCH and other algorithms mentioned above are all unsupervised clustering algorithms and do not use any labels, even when they are available. All processing, including sorting, alignment and clustering, are done using the raw sequences only.

III. Methods

A. Incremental VSEARCH

We first described incremental VSEARCH (I-VSEARCH) in [12] primarily to reduce the computational cost of consecutive clustering processes when new batches of data arrive, while maintaining similar clustering performance. I-VSEARCH takes advantage of VSEARCH's algorithmic design decision that only the seeds are used for computing the similarity metric during clustering, while other sequences are no longer used once they are placed in their respective clusters.

I-VSEARCH is a wrapper for VSEARCH. For the first batch of data received, I-VSEARCH first calls VSEARCH to have the data sorted, aligned and clustered. Once the clustering is complete, I-VSEARCH saves only the seeds of each cluster in preparation to receive the next batch of data. When the next batch of data is received, this batch is sorted based on length, but seeds saved from the previous batch(es) are added to the beginning of the new data file to ensure that the same clusters created previously are recreated quickly before any new data is processed. If the same similarity threshold is chosen, I-VSEARCH replicates the same clusters by processing only those seeds. Once the clusters from previous run are recreated, regular clustering proceeds by processing new sequences according to the same rules as before. After each batch of data, the seeds are reassigned in case of any cluster receives a longer sequence than its prior seed.

B. Semi-Supervised Incremental VSEARCH (SSI-VSEARCH)

We now focus on the SSL to determine the classification capabilities and properties of our approach through a comprehensive set of experiments. To do so, we describe Semi-Supervised Incremental VSEARCH (SSI-VSEARCH), which transforms the clustering algorithm to a semi-supervised classification algorithm by taking advantage of any – however small – labeled reference dataset available, with the additional benefit of doing so incrementally.

SSI-VSEARCH is essentially another wrapper, this time around I-VSEARCH, that adds the label information to the unlabeled sequence. Here, we assume that a small portion of the sequences do in fact have known labels, or there is a reference database available that consists of labeled sequences, some of which may or may not appear in the query dataset. For the purposes of this discussion, we refer to sequences that have known labels as *known sequences*, and those that do not have labels as *unknown sequences*. We will also refer to *novel sequences*, those that have never been seen before and has never been labeled in any reference dataset.

In SSI-VSEARCH, the first step is to run the original I-VSEARCH on the first batch of data. For this first step, the labels of the known sequences are detached from the sequences, and they are treated as if they are unlabeled. After the clustering is completed, the labels of the known sequences are re-attached to the data structure that holds the sequences. At this stage, the algorithms faces three possible scenarios for each cluster:

- The cluster includes one or more labeled sequences of the same label and possibly one or more unlabeled sequences. In this case, the cluster - and all unlabeled sequences within it - are given the known label. Such a cluster is deemed *pure*, as all of its sequences have the same label.
- The cluster includes multiple labeled sequences – but of different labels – as well as unlabeled sequences. In this case, the cluster is assigned the label that is determined through a majority vote process. The label associated with the largest number of (known) sequences in the cluster is the winning label, and that label is then attached to all unlabeled sequences. Ties can be broken randomly, or by accepting the label of the longest known sequence as the winning label. Such a cluster is an impure cluster, since it includes sequences from at least one additional label. In both this and the previous case, labels attached to a previously unlabeled/unknown sequence, are referred to as *predicted labels* and the corresponding sequence is considered a *predicted sequence*.
- The cluster does not include any labeled (known) sequence. Then, the cluster – and all unknown / unlabeled sequences in it – are given a random temporary label.

The incremental learning ability of the algorithm may add sequences to any of the existing clusters created in previous runs on earlier batches of data. In such cases, the same procedure described above is repeated. Of particular importance is the last scenario: if a known sequence is later added to a previously unknown cluster with a temporary label, the cluster and the sequences within the cluster are labeled according to the majority vote scenario described above.

C. Evaluation

We define several new metrics to evaluate the algorithm's performance and behavior, as described below. To determine the true ability of the algorithm to predict correct labels, the ground truth, i.e., the actual correct labels are needed. Therefore, we used the 16S rRNA dataset (described in more detail below), for which all labels are in fact available to us. To simulate reference data of known labels and experimental data of unknown labels, we partitioned the entire dataset into training and testing subsets, respectively. The training data subset (simulating the reference dataset) consisted of 25% of the entire dataset, whereas the remaining 75% was set aside as test data (simulating experimental/environmental data). The labels of training dataset were provided to the algorithm *after* the clustering stage to be used during majority voting, whereas the labels of the test data were kept hidden from the algorithm. The labels of the test data were only used later as ground truth (for computing various evaluation metrics) described below.

1) Unlabeled Cluster and Sequence: After the majority voting, all sequences are assigned a label per the process described above. Those sequences (and clusters) that do not have a known label are assigned a temporary ID, but are still considered *unlabeled* for evaluation and classification purposes. Naturally, there is no actual prediction made for these sequences. Hence unlabeled sequences are not counted

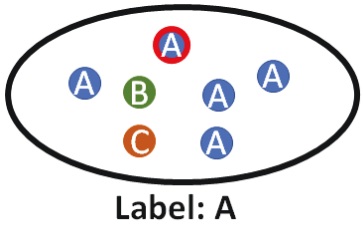

Fig. 1. In given cluster, assume that red outlined sample A comes from training/reference dataset, and hence has a known label. After majority voting using reference samples, cluster is labeled as A. While calculating predicted accuracy, we only consider samples from testing dataset (4A, 1B, 1C). Accuracy for this cluster is then 4/6.

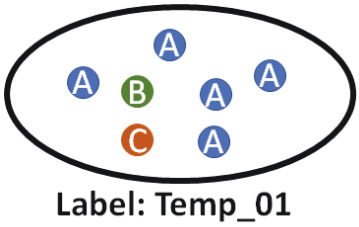

Fig. 2. All samples in this unlabeled cluster are from the test data and a temporary label is assigned to the cluster. While calculating unlabeled accuracy, we check ground truth and assign cluster label accordingly. Here, let's assume that the true labels are A, B and C as shown. Given that the cluster has 5As, 1B and 1C, unlabeled accuracy is 5/7.

when computing accuracy since – in the absence of ground truth – accuracy cannot be determined. These sequences are counted, however, in determining the prediction rate of the algorithm (defined below).

2) Labeled Cluster and Predicted Sequences: When a cluster has at least one sample from reference dataset, that cluster and its unlabeled sequences are labeled based on the majority voting of the labels of known sequences. Such a cluster is considered as a labeled cluster. All such sequences from previously unknown experimental test data are then considered as *predicted sequences*.

3) Prediction Accuracy: Prediction accuracy is the ratio of the predicted sequences that are actually given the correct label by the algorithm, as compared to the ground truth. Prediction accuracy is only computed on the test dataset so that it is not artificially increased with labels of the training data that are already made available to the algorithm during the majority voting stage. Prediction accuracy is computed simply as $PA = PS_{correct}/(PS_{correct}+PS_{incorrect})$ where, $PS_{correct}$ and $PS_{incorrect}$ are the number of predicted sequences in the test dataset whose labels are determined to be correct and incorrect, respectively, based on ground truth data.

4) Prediction Rate: While we cannot assign an accuracy metric to sequences that are left unlabeled (but only given a temporary ID), we do keep track of how many such sequences are left in the dataset. We monitor them through the *prediction rate* metric, which is the ratio of number of predicted sequences to the total number of sequences (predicted and unlabeled) in the dataset.

5) Unlabeled Accuracy: Recall that unlabeled clusters are those that do not have any known reference data label, and hence are provided a temporary ID by the algorithm. In a real world scenario where the sequences in such clusters are truly unknown, it is impossible to know whether these sequences even belong to the same class. Nevertheless, since all labels of the RDP dataset are known to us, we can in fact compute the accuracy of even those sequences. We keep a separate metric for such sequences, referred to as *unlabeled accuracy* UA, distinct from prediction accuracy described above, and compute it as $UA = US_{correct}/(US_{correct} + US_{incorrect})$ where, $US_{correct}$ and $US_{incorrect}$ are the number of test data sequences left unlabeled after the majority voting whose labels are determined to be correct and incorrect, respectively, compared to ground truth data. Figure 2 provides a graphical interpretation of this metric.

6) Completeness and Homogeneity: Completeness and homogeneity are popular metrics used in analysis of genomic sequences. They are both information theoretic metrics, based on entropy, and are normalized to have a value between 0 and 1. Completeness measures the degree to which members of a particular label are all in one cluster. Completeness can be calculated as

$$Completeness = 1 - \frac{H(C|L)}{H(C)} \tag{1}$$

$$H(C|L) = -\sum_{c=1}^{C}\sum_{l=1}^{L}\frac{n_{c,l}}{n} log\left(\frac{n_{c,l}}{n_l}\right) \tag{2}$$

$$H(C) = -\sum_{c=1}^{C}\frac{n_c}{n} log\left(\frac{n_c}{n}\right) \tag{3}$$

where $H(\cdot)$ and $H(\cdot|\cdot)$ are entropy and conditional entropy, respectively, l is an index on labels with L as the total number of labels, c is an index on clusters with C as the total number of clusters, n_c is the number of samples in cluster c, $n_{l,c}$ is the number of samples with label l in cluster c, and n is the total number of samples in the dataset.

Homogeneity, on the other hand, is a measure of purity of clusters. If all clusters are pure, i.e., each cluster contains members of the same class label, and none of the other labels, then the homogeneity is 1 for that clustering. Homogeneity can be calculated as

$$Homogeneity = 1 - \frac{H(L|C)}{H(L)} \tag{4}$$

$$H(L|C) = -\sum_{l=1}^{L}\sum_{c=1}^{C}\frac{n_{l,c}}{n} log\left(\frac{n_{l,c}}{n_c}\right) \tag{5}$$

$$H(L) = -\sum_{l=1}^{L}\frac{n_l}{n} log\left(\frac{n_l}{n}\right) \tag{6}$$

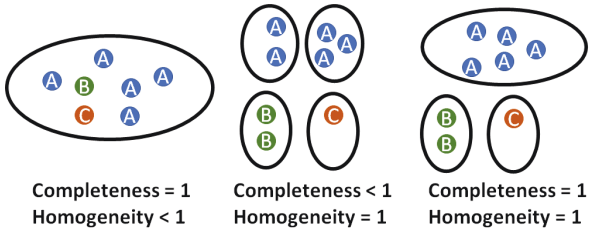

Fig. 3. In the first case, there is only one cluster, so completeness is 1 by definition, however homogeneity is less than 1. In the middle case, homogeneity is 1 since each of the clusters is pure, but completeness is less than 1 since not all "A" labels are in one cluster. In the third case, both completeness and homogeneity are 1 since all clusters are pure, and all sequences of a given label are in one cluster each.

where previously mentioned variables are the same (with $n_{c,l} = n_{l,c}$) and n_l is the number of samples with label l. Figure 3 illustrates completeness and homogeneity.

7) V Measure: V Measure is a composite metric that combines completeness and homogeneity (through geometric mean) to provide an overall assessment of the clustering algorithm. V-measure is computed as

$$VMeasure = \frac{(1+\beta) * Homogeneity * Completeness}{\beta * Homogeneity + Completeness} \quad (7)$$

where β weighs the relative importance of homogeneity over completeness. We used $\beta = 1$ to keep the relative importance of completeness and homogeneity the same.

8) Singletons: Singletons are those sequences that are left alone and are not clustered with any other sequence. Singletons, whether labeled or otherwise, are not included in accuracy calculations to keep these metrics as fair as possible.

IV. Experimental Setup

A. RDP18 Dataset

RDP18 is the 18^{th} and the most recent release (in 2020) of the 16S rRNA dataset, provided by the Ribosomal Database Project [5]. RDP18 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). 25% of the dataset was used for *training* where labels were provided to the algorithm at the majority-vote stage, and the remaining 75% set aside for *testing* (or prediction) whose labels were not provided to the algorithm. We removed 6 phyla that had fewer than four total samples. The final dataset had 20,186 samples from 32 phyla, divided into training (5,035) and testing (15,151) datasets. Training dataset is further divided into five batches to simulate an incremental learning setting using two experimental protocols.

B. Experiments

In our first experiment, the five training batches are created by assigning samples to each batches completely at random. This sampling results in each batch having a similar distribution as the original dataset. We refer to this experiment as the *random sampling (RS) experiment*, since the training data subsets used for each batch are randomly sampled from the original training dataset, following the same distribution of the original data. The algorithm is incrementally run with consecutively using the five batches of training subsets, updating the clusters after each batch. After each update, we evaluate the performance of SSI-VSEARCH on the test dataset.

For the second experiment, the training batches are curated such that phyla are mutually exclusive to batches. This scenario simulates a more challenging setting, where the algorithm can only see samples of any given label only once. We refer to this second experiment as the *incremental phyla (IP) experiment*, where each batch of data includes a different (mutually exclusive) set of phyla. There are 32 phyla in the dataset, but four of these phyla are dominant and compose 90.42% of the dataset. Sequences from these four dominant phyla are used exclusively in the first four batches, and the final batch included samples from all remaining 28 phyla. In this experiment, novel information from new phyla are introduced incrementally to test the ability of the algorithm to learn new classes one at a time.

The primary free parameter of SSI-VSEARCH is the similarity threshold. In general, lower thresholds are used to analyze the sequences at more general taxonomic ranks (such as phyla), whereas higher thresholds are used for analyzing more specific taxonomic ranks (such as genus). For completeness, we run all experiments on all thresholds from 75% to 97% at all taxonomic ranks.

We used prediction accuracy, prediction rate and unlabeled accuracy to evaluate the classification performances of SSI-VSEARCH. We also compared SSI-VSEARCH to the original VSEARCH on its clustering capabilities. Note that VSEARCH processes all available data at once, and hence has the luxury of using all sequences to determine the clusters, whereas SSI-VSEARCH sees only a subset of the sequences at a time. We then use completeness, homogeneity and V-Measure to compare clustering performance and time consumption to compare efficiency of these two methods.

V. Results and Discussion

A. SSI-VSEARCH vs. VSEARCH on Clustering Performance

We first compare SSI-VSEARCH and VSEARCH performance with respect to completeness, homogeneity and v-measure on all similarity thresholds. These results represent the quantities as computed after SSI-VSEARCH completed its five incremental batches, and after VSEARCH completed its single run on the entire dataset.

Figures 4 and 5 compare the clustering metrics for VSEARCH vs. SSI-VSEARCH for both random sampling and incremental phyla experiments, respectively, at the genus level. The primary takeaway from these figures is that SSI-VSEARCH does not lose any performance despite processing the data incrementally, seeing only some of the data at a time, while VSEARCH has the luxury of having access to all of the data at once. This observation is all the more remarkable considering that SSI-VSEARCH only carries the cluster seeds

from one batch to the other, and discards all other – what appears to be *nonessential* – data.

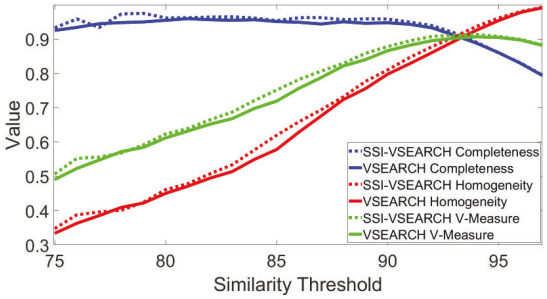

Fig. 4. Comparison of VSEARCH and SSI-VSEARCH for completeness, homogeneity and v-measure at similarity thresholds 75-97 and genus level for random sampling experiment.

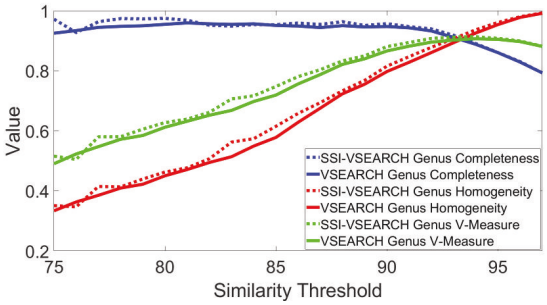

Fig. 5. Comparison of VSEARCH and SSI-VSEARCH using completeness, homogeneity and v-measure at similarity thresholds 75-97 and genus level for incremental phyla experiment.

We note that – while we did compute these quantities for all taxonomic ranks, we report the results only for the genus level for brevity, which are the most conservative results. The comparisons at other levels were similar (with no difference between SSI-VSEARCH and VSEARCH), with the absolute results for more general taxonomic ranks in fact being higher for each of the metrics.

We also note that carrying only the seed information from one batch to the next and removal of non-essential data saves space and speeds up the clustering process. If run on combined new and old data, VSEARCH would recluster the same (old) data over and over again, a step avoided by SSI-VSEARCH. Fig. 6 shows the percentage of time saved by SSI-VSEARCH over VSEARCH when clustering incrementally. Figure 6 shows that SSI-VSEARCH saves as much as 60-70% time over VSEARCH at lower thresholds, though relative time saved at higher thresholds is less. This is because, it is harder to cluster samples together and there are significantly more clusters at higher similarity thresholds. Therefore, more cluster seeds must be transferred to the next step, which means only a smaller part of the data is nonessential and hence less data are discarded. For lower similarity thresholds, significantly more samples are deemed nonessential and discarded after first

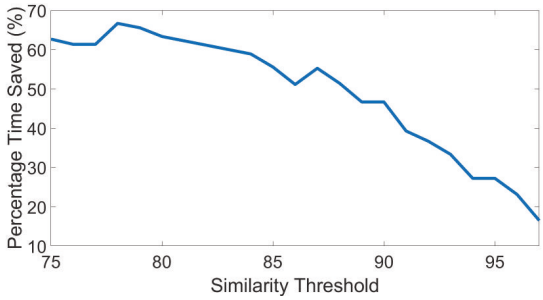

Fig. 6. Percentage time saved by using SSI-VSEARCH instead of VSEARCH for clustering the same data.

processing by SSI-VSEARCH. Regardless of the similarity threshold, however, SSI-VSEARCH always saves time without losing any performance on any of the evaluation metrics.

B. Prediction Capabilities of SSI-VSEARCH

As described in previous sections, a significant advantage of SSI-VSEARCH over VSEARCH is its incremental and semi supervised training, which allows it to make a classification prediction and provide labels to unknown samples by processing the data one batch at a time. Recall that only labels of the training data are provided to the algorithm during the majority voting stage, whereas the algorithm is asked to predict the labels of the test data, whose labels are not shown to the algorithm. The ground truth labels for the test data are only used later during evaluation for computing prediction accuracy. For these set of experiments, we therefore focus on the classification performance and other incremental learning properties of SSI-VSEARCH. Specifically, we provide prediction accuracy, prediction rate, and unlabeled accuracy of the algorithm at all similarity thresholds and for all taxonomic depths (ranks).

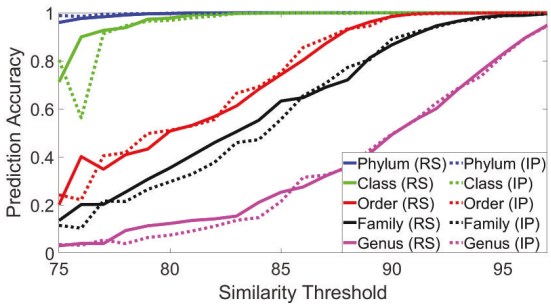

Fig. 7. Prediction accuracy of SSI-VSEARCH at every similarity threshold and taxonomic level in random sampling (solid lines) and incremental phyla (dotted lines) experiments.

We start with prediction accuracy of SSI-VSEARCH, for both the random sampling (RS) and incremental phyla (IP) experiments, which are shown in Figure 7. As expected, the prediction accuracy is higher for higher similarity thresholds and for more general taxonomic ranks (such as phylum).

Also as expected, the prediction accuracy is lower for more specific taxonomic ranks (such as genus) particularly at lower similarity thresholds, but improves dramatically towards 100% at higher similarity thresholds. We observe that the prediction accuracy is effectively 100% for phylum, class, and order at 90% or higher similarity thresholds. For genus rank, appropriate similarity threshold is considered to be over 95% and prediction accuracy for genus in that range is around 80%.

Another interesting metric to consider for SSI-VSEARCH is the *prediction rate*, percentage of unknown sequences that the algorithm can actually predict. Recall that not all sequences can be labeled. Specifically, sequences that end up in a cluster that does not include any known labels cannot be classified. Fig. 8 shows the prediction rate of the algorithm (on the random sampling experiment) for each similarity threshold after each batch of incremental learning. We observe that the percentage of the sequences that can be predicted increases from Batch 1 (blue) to Batch 5 (orange), as we would hope and expect from an incremental learning algorithm. This observation demonstrates that the algorithm is indeed learning new information with each new batch, and predict labels of previously unknown sequences. We also see that – again as expected – the prediction rate is higher for lower similarity thresholds. At a typical threshold of 90%, the algorithm classifies about 90% of the unknown test sequences after completing its training on the 5^{th} batch (orange curve).

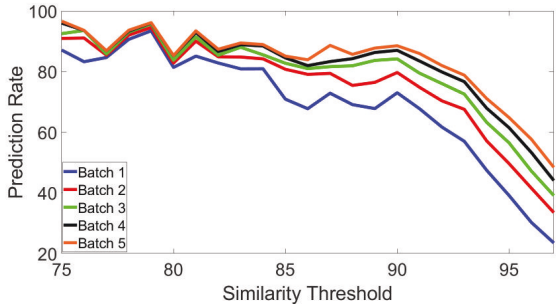

Fig. 8. Prediction rate of SSI-VSEARCH at every similarity threshold after every batch in random sampling experiment.

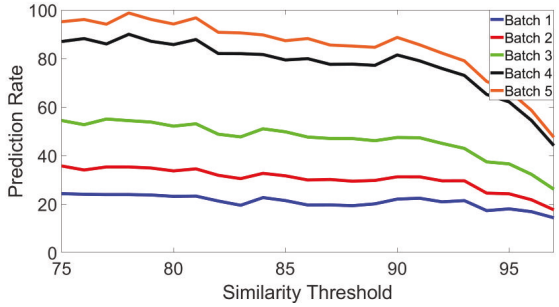

Fig. 9. Prediction rate of SSI-VSEARCH at every similarity threshold after every batch in incremental phyla experiment.

The incremental phyla experiment more clearly demonstrates the incremental learning ability of the algorithm. Fig. 9 plots the prediction rate of SSI-VSEARCH in incremental phyla experiment, which shows distinct differences from that of random sampling experiment in Fig. 8. Recall that in random sampling experiment, there are examples of all phyla even in the first batch, and each batch that comes after that. Therefore, there is ample information for majority of samples to be predicted even after the first batch, with modest additional new information provided with each subsequent batch. As a result, the initial prediction rate is much higher after the first batch for the random sampling experiment compared to incremental phyla experiment (blue curves in Figures 8 and 9). For the same reason, there are relatively modest additional prediction rate improvements after each subsequent batch from Batch 1 through Batch 5 in Fig. 8. When different phyla are exclusive to each batch in the incremental phyla experiment, however, the initial prediction rate is only around 20% after the first batch, which makes sense since the algorithm has seen only about a fifth of all available labels. With this experiment, there is considerably more – and novel – information provided to the algorithm by each batch, and the algorithm does indeed learn more with each batch as demonstrated by the more significant jumps in prediction rate from each batch to the next as seen in Fig. 9.

It is also interesting to note that adding new information phylum by phylum versus adding a mixture of phyla at each batch does not actually change the final prediction rate of the algorithm. We observe from Figures 8 and 9 that the prediction rates after Batch 5 (orange curves) are similar to each other.

We note that some of the unlabeled sequences are left unlabeled because they end up as singletons, i.e., clusters of one sequence. The number and percentage of sequences left as singletons increase with the similarity threshold. This is also expected: with increasing similarity threshold, the algorithm requires more overlap in sequences before they can be grouped into the same cluster. We observed that about 5000 sequences were left as singletons at 97% similarity, which constitutes about 30% of the total test data. At the more typical 93% similarity threshold, about only 1000 singletons (about 6.6% of the total test data) were left unclassified, and about 25% of all unlabeled sequences were singletons.

Singletons and other unlabeled clusters are not included in our prediction accuracy results as they do not have labels. However, since the ground truth labels of all sequences are known in the RDP dataset, we can in fact calculate the accuracy of unlabeled clusters. We refer to this metric as *unlabeled accuracy* to keep it distinct from *prediction accuracy* as described above. Figures 10 and 11 show the unlabeled accuracy of SSI-VSEARCH in both random sampling and incremental phyla experiments at every similarity threshold and taxonomic rank. The unlabeled accuracy is similar to – or even better than – the prediction accuracy (in Fig. 7), which indicates that SSI-VSEARCH actually clusters the unknown sequences accurately while waiting for additional label information from a future reference dataset. Accordingly,

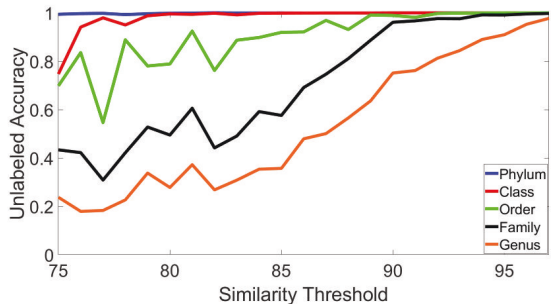

Fig. 10. Unlabeled accuracy of SSI-VSEARCH at every similarity threshold and taxonomic level for random sampling experiment.

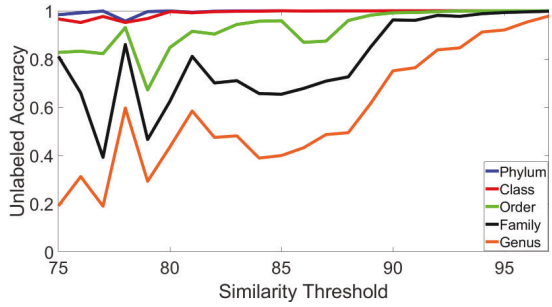

Fig. 11. Unlabeled accuracy of SSI-VSEARCH at every similarity threshold and taxonomic level for incremental phyla experiment.

as information about as-yet-unlabeled sequences does become available in the future, the algorithm will be able to correctly identify previously unlabeled novel sequences with the correct label. This observation is one of the most reassuring aspects of SSI-VSEARCH in its ability to identify novel sequences through incremental processing.

VI. Conclusions

In this paper, we demonstrated the additional capabilities of SSI-VSEARCH over the original VSEARCH, while also illustrating its behaviour and properties over several carefully designed experiments on the RDP18 16S-rRNA dataset. We showed that through a semi-supervised approach, SSI-VSEARCH converts VSEARCH – an otherwise purely clustering algorithm that can only group similar sequences in clusters – to a classification algorithm that can predict labels of unknown sequences with high accuracy at all taxonomic levels. The algorithm can do so whether the sequences are introduced randomly or one/few phyla at a time. Furthermore, SSI-VSEARCH can incrementally process additional data that later become available without access to – or reprocessing – the old data, saving computational time without any loss on clustering based metrics compared to VSEARCH. We have also shown that clusters that SSI-VSEARCH leaves unlabeled, i.e., the sequences in unlabeled clusters, can be predicted accurately once the necessary label information is added to a reference database.

Our immediate future work primarily includes expanding the experimental analysis to larger datasets, such as the 9^{+} million sequence SILVA database, as well as protein datasets to make sure that the algorithm is scalable and versatile. Future work also includes integrating the general approach of adding semi-supervised incremental learning capability to bioinformatics platforms and pipelines other than VSEARCH, such as MG-RAST and MMseqs2. Given that most platforms generally use similar clustering approaches, those platforms may also benefit from additional semi-supervised classification capabilities with suitable modifications. Finally, we will also explore mechanisms that can be added to the algorithm to further minimize, or properly combine, the singletons or other unlabeled clusters into (larger) clusters when appropriate.

References

[1] 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.
[2] B. Buchfink, K. Reuter, and H. Drost, “Sensitive protein alignments at tree-of-life scale using DIAMOND,” Nature Methods, vol. 18, no. 4, pp. 366–368, 2021.
[3] B. Buchfink, C. Xie, D. H. Huson, “Fast and sensitive protein alignment using DIAMOND,” Nature Methods, vol. 12, no. 1, pp. 59–60, 2015.
[4] H. Chai, Y. Liang, S. Wang, and H. Shen, “A novel logistic regression model combining semi-supervised learning and active learning for disease classification,” Scientific Reports, vol. 8, no. 1, pp. 1–10, 2018.
[5] 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.
[6] R. C. Edgar, “Search and clustering orders of magnitude faster than BLAST,” Bioinformatics, vol. 26, no. 19, pp. 2460–2461, 2010.
[7] M. Hauser, C. E. Mayer, and J. Söding, “kClust: fast and sensitive clustering of large protein sequence databases,” BMC Bioinformatics, vol. 14, no. 1, pp. 1–12, 2013.
[8] 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.
[9] 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.
[10] 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.
[11] M. Mirdita, M. Steinegger, F. Breitwieser, et al. “Fast and sensitive taxonomic assignment to metagenomic contigs,” Bioinformatics, vol. 37, no. 18, pp. 3029–3031, 2021.
[12] 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.
[13] T. Rognes, T. Flouri, B. Nichols, et al. “VSEARCH: a versatile open source tool for metagenomics,” PeerJ, vol. 4, pp. e2584, 2016.
[14] S. L. Westcott and P. D. Schloss, “De novo clustering methods outperform reference-based methods for assigning 16S rRNA gene sequences to operational taxonomic units,” PeerJ, vol. 3, pp. e1487, 2015.
[15] C. R. Woese and G. E. Fox, “Phylogenetic structure of the prokaryotic domain: the primary kingdoms,” Proceedings of the National Academy of Sciences, vol. 74, no. 11, pp. 5088–5090, 1977.
[16] 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.