Incremental and Semi-Supervised Learning of 16S-rRNA Genes For Taxonomic Classification

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

Norman C. Sabin Jr.
Electrical and Computer Eng.
Rowan University
Glassboro, NJ, USA
sabinn49@students.rowan.edu

Thomas Gracie III
Electrical and Computer Eng.
Rowan University
Glassboro, NJ, USA
tom.gracie.iii@gmail.com

Steven Portley
Electrical and Computer Eng.
Rowan University
Glassboro, NJ, USA
portleys4@gmail.com

Mali Halac
Electrical and Computer Eng.
Drexel University
Philadelphia, PA, USA
mh3636@drexel.edu

Thomas Coard
Electrical and Computer Eng.
Drexel University
Philadelphia, PA, USA
tgc37@drexel.edu

William Trimble
Argonne National Labs
University of Chicago
Chicago, IL, USA
wltrimbl@uchicago.edu

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

2021 IEEE Symposium Series on Computational Intelligence (SSCI) | 978-1-7281-9048-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/SSCI50451.2021.9660093

***Abstract*—Genome sequencing generates large volumes of data and hence requires increasingly higher computational resources. The growing data problem is even more acute in metagenomics applications, where data from an environmental sample include many organisms instead of just one for the common single organism sequencing. Traditional taxonomic classification and clustering approaches and platforms – while designed to be computationally efficient – are not capable of incrementally updating a previously trained system when new data arrive, which then requires complete re-training with the augmented (old plus new) data. Such complete retraining is inefficient and leads to poor utilization of computational resources. An ability to update a classification system with only new data offers a much lower run-time as new data are presented, and does not require the approach to be re-trained on the entire previous dataset. In this paper, we propose Incremental VSEARCH (I-VSEARCH) and its semi-supervised version for taxonomic classification, as well as a threshold independent VSEARCH (TI-VSEARCH) as wrappers around VSEARCH, a well-established (unsupervised) clustering algorithm for metagenomics. We show – on a 16S rRNA gene dataset – that I-VSEARCH, running incrementally only on the new batches of data that become available over time, does not lose any accuracy over VSEARCH that runs on the full data, while providing attractive computational benefits.**

***Index Terms*—Incremental clustering, taxonomic classification, VSEARCH, 16S rRNA genes**

I. Introduction

Genetic sequencing involves reading and recording of the base units of biological materials such as DNA, RNA, and proteins. It is often used to identify and gain information regarding the organisms being sequenced. The most common methods of sequencing involve growing a single organism in a mono-culture before extracting genetic materials [1]. This process introduces limitations to both the types of organisms that can be sequenced and the information that can be derived from the sequencing.

Metagenomics is a field based on methods that involve the genetic sequencing of environmental samples, which typically have large numbers of different organisms within them. Metagenomics provide greater information related to the interaction between organisms and their environment. Metagenomics has many practical applications, such as agriculture and the study of gut microbiomes. Some crop responses to disease, for example, could only be understood when the present microorganisms are considered together, rather than in isolation [1]. The same principles can be applied to the bacteria communities living in the human gut, where complex cross-species interactions can only be understood in the wider context of the microbial environment.

Due to the heterogeneous nature of metagenomic samples, classification algorithms are often used to predict the taxonomy of newly sequenced genetic material. Because of

This work is supported by National Science Foundation grant #1936782.

978-1-7281-9048-8/21/$31.00 ©2021 IEEE

Authorized licensed use limited to: Rowan University Libraries. Downloaded on August 29,2022 at 05:33:32 UTC from IEEE Xplore. Restrictions apply.

the large volumes of data often produced in metagenomic sampling, however, the computational costs of training these algorithms can be extensive. The computational cost can be particularly – and unnecessarily – high when algorithms must be entirely retrained from scratch with all data accumulated thus far just to incorporate new data, even if such new data come from only a small number of organisms and resulting in just a minuscule percentage of the previously generated dataset. Incremental learning, which allows an algorithm to adapt to the new information by training only on the new data, can be useful in this context. In this proof-of-concept effort, we introduce I-VSEARCH, an incrementalized version of VSEARCH, a popular open source metagenomics tool for alignment and clustering of nucleotide sequence data, such as the 16S rRNA genes.

II. Background

A. Genomic Clustering Algorithms

We start with CD-Hit, a fast comparison and clustering algorithm for nucleotide and protein sequences. CD-Hit is a greedy algorithm that starts by comparing the longest sequence to an initially empty, but increasingly sorted collection of *seeds* that represent the centroid of a cluster. If the query sequence is similar enough (with respect to overlapping number of nucleotides) to a seed by common word count, then the sequence is placed within the cluster that the seed is representing, otherwise it becomes the seed for a new cluster [2]. USEARCH is a tool similar to CD-Hit that provides advantages such as higher speeds and less memory usage. It is able to do this by first sorting based upon similar word count, rather than length, and terminating early knowing that the chance for a match rapidly drops as you continue [3]. VSEARCH, the algorithm used in this paper, is a freely available and open-source version of USEARCH. VSEARCH is very similar to USEARCH, except that it "performs optimal global sequence alignment of the query against potential target sequences, using full dynamic programming instead of the seed-and-extend heuristic used by USEARCH" [4], and contains other metagenomic tools such as chimera detection and dereplication[1].

There are other platforms, particularly well-suited for metagenomics applications, such as MG-RAST[2] [5] and DIAMOND[3] [6], both of which use a similar clustering and alignment procedures as VSEARCH, though they are primarily intended for protein sequences. While in this proof-of-concept work, we focus on 16S rRNA nucleotide data using VSEARCH, both MG-RAST and DIAMOND can also benefit from the concepts introduced here, as neither is capable of processing data incrementally.

B. VSEARCH Clustering

VSEARCH primarily consists of two steps, a pre-processing step that includes sorting and alignment, followed by a clustering step.

[1]VSEARCH can be downloaded from https://github.com/torognes/vsearch
[2]MG-RAST is available at https://www.mg-rast.org/
[3]DIAMOND can be downloaded at https://github.com/bbuchfink/diamond

In the pre-processing (sorting and alignment) step, all sequences are first sorted from longest to shortest based on sequence length. Then, sequences are aligned to evaluate their similarities, using the Needleman-Wunsch global alignment algorithm. Needleman-Wunsch method equalizes lengths of the sequences by carefully placing gaps to help align similar sections of different sequences. In the clustering step, once all sequences are sorted and aligned, VSEARCH compares each sequence to the representative sequence, or *the seed*, of each cluster and computes a similarity score. VSEARCH starts with an initially empty list of clusters, and creates new clusters as needed. The algorithm's primary free parameter is its similarity threshold, used to determine when a new cluster is to be created. For each query sequence, if it finds a cluster whose seed is similar enough to exceed the similarity threshold, then the query sequence is added to that cluster. If such a cluster does not exist, then the query sequence becomes the seed of a new cluster and is added to the list of clusters. In other words, the first sequence, by definition the longest in length after sorting, is considered the seed of the first cluster. The next sequence is compared to this seed. If the similarity score of the second sequence and the seed is higher than the similarity threshold, it joins the first cluster, otherwise, it becomes the seed of its own new cluster. The next sequence is compared to the existing cluster seeds, and the process is repeated until all sequences have been clustered. Pseudo-code for this process is given in Algorithm 1. A toy example is illustrated in Fig.1. It is important to note that only the sequences themselves are used for clustering, and the labels for those sequences – even if they are available – are not used. This process is therefore completely unsupervised; the sequence ID, taxonomic ranks or any other label that may be available are not used by VSEARCH. It is also important to note that sequences are not compared to all cluster seeds. As soon as a query sequence finds a cluster seed that is within the similarity threshold, the searching stops and the query sequence joins that cluster - such an approach considerably enhances the run-time speed of VSEARCH over other clustering algorithms at the cost of a possible small drop in accuracy.

III. Procedure

We have developed three different modifications to the original VSEARCH algorithm to address three common problems, namely, allowing the algorithm to learn incrementally, allowing the algorithm to learn labels in a semi-supervised manner, and allowing the algorithm to perform even when the optimal similarity threshold is unknown. These algorithms are described below.

A. I-VSEARCH: Incremental VSEARCH

Looking at the VSEARCH algorithm processes in detail, we noticed that – while the algorithm itself is not capable of, nor was intended for incremental learning – its underlying clustering algorithm can be easily extended to incremental processing with suitable modifications. This conclusion is based on a fundamental observation of VSEARCH: when

Authorized licensed use limited to: Rowan University Libraries. Downloaded on August 29,2022 at 05:33:32 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1 VSEARCH algorithm

Inputs: $\mathcal{D}$: dataset consists of sequences $\{\mathcal{S}_1, \ldots, \mathcal{S}_J\}, j \in J$; ϕ: Threshold value

$\mathcal{C}, \mathcal{X} \leftarrow$ Initialize empty cluster set and cluster seed set.
$\mathcal{D} \overleftarrow{sort}\ \mathcal{D}$ (Sort sequences from longest to shortest)
for each sequence $j \in \mathcal{D}$ **do**
  for each cluster $i \in \mathcal{C}$ **do**
    $similarity = f(\mathcal{S}_j, \mathcal{X}_i)$ (Calculate similarity between sequence and cluster seed)
    if $similarity > \phi$ **then**
      Sequence $\mathcal{S}_j$ joins $\mathcal{C}_i$
      break
    else
      Sequence $\mathcal{S}_j$ creates a new cluster $\mathcal{C}_{new}$
      $\mathcal{X}_{new} \leftarrow \mathcal{S}_j$ (Sequence becomes the seed of the cluster)
      break
    end if
  end for
end for

Output: $\mathcal{C}$, cluster set, $\mathcal{X}$, set of cluster seeds

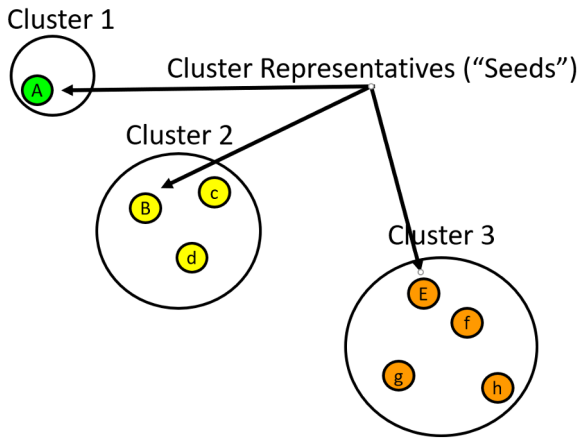

Fig. 1. VSEARCH clustering is entirely unsupervised. Here, larger circles represent clusters, each having at least a seed – indicated in capital letters as (A), (B) and (E) – and possibly other sequences – indicated in lower case letters as (c), (d), (f), (g), and (h).

a query sequence searches for a cluster to join, it is only compared to the seeds of each cluster; the other sequences that have previously joined that cluster are never used, processed or seen by the algorithm, a deliberate design decision that provides significant computational savings. However, this observation also means that future runs of VSEARCH only need to use the seeds of the clusters to be able to pick up where the previous run left off, so long as some additional bookkeeping is maintained. Giving VSEARCH the capability to learn incrementally primarily requires storing, restructuring and then reordering of its previously determined cluster seeds. We refer to our modified algorithm that has this incremental learning capability as Incremental VSEARCH, or simply as I-VSEARCH.

Given a set of batches of data that we wish to cluster incrementally using I-VSEARCH, we process batches one by one. First, the current batch is sorted from longest to shortest, and sequences are clustered using standard VSEARCH. Then, after each batch, the seeds of clusters are saved to transfer current state of knowledge to the next step. These seed sequences from the previous batch are added to the *beginning* of the sorted next batch before processing. Since VSEARCH processes each sequence sequentially, it can be guaranteed that the seeds from the prior batch will be retained as seeds in the next run of incremental I-VSEARCH. The algorithm can then continue clustering from where it left off, as new data come in, without needing to process the entire old data.

Algorithm 2 Incremental VSEARCH algorithm

Inputs: $\mathcal{B} = \mathcal{B}_1, \ldots, \mathcal{B}_K$: set of batches. Every batch $\mathcal{B}_k$ consists of sequences $\{\mathcal{S}_{k1}, \ldots, \mathcal{S}_{kJ}\}, k \in K, j \in J$; $\mathcal{X}_{old}$: Previous cluster seeds(sorted); ϕ: Similarity threshold

for each batch $k \in K$ **do**
  $\mathcal{C}, \mathcal{X} \leftarrow$ Initialize empty cluster set and cluster seed set.
  $\mathcal{B}_k \overleftarrow{sort}\ \mathcal{B}_k$ (Sort sequences from longest to shortest)
  if $\sim$ isempty($\mathcal{X}_{old}$) **then**
    $\mathcal{B}_k \leftarrow [\mathcal{X}_{old}; \mathcal{B}_k]$ (If cluster seeds from previous batch exist prepend them to the new batch)
  end if
  for each sequence $j \in \mathcal{B}_k$ **do**
    for each cluster $i \in \mathcal{C}$ **do**
      $similarity = \min_{i \in I} f(\mathcal{S}_{kj}, \mathcal{X}_i)$ (Calculate similarity between sequence and cluster seed)
      if $similarity > \phi$ **then**
        Sequence $\mathcal{S}_{kj}$ joins $\mathcal{C}_i$
        break
      else
        Sequence $\mathcal{S}_{kj}$ creates a new cluster $\mathcal{C}_{new}$
        $\mathcal{X}_{new} \leftarrow \mathcal{S}_{kj}$ (Sequence becomes the seed of the cluster)
        break
      end if
    end for
  end for
  $\mathcal{X}_{old} \leftarrow \mathcal{X}$
end for

Output: $\mathcal{C}$, cluster set, $\mathcal{X}$, set of cluster seeds

In terms of computational savings, the ability to run VSEARCH incrementally eliminates the need to run the algorithm repeatedly on the same data, and provides considerable savings over time. To illustrate, let us assume that running the algorithm on a dataset of N sequences takes T seconds (where we use time as a proxy to actual number of operations), and later we receive a new dataset that also includes N sequences. Even if the complexity of the alignment / clustering algorithm were linear, i.e., $\mathcal{O}(N)$, total run time using single batch processing would be $3T$ (T for the first run, $2T$ for the second run with the newly expanded dataset). Using the incremental

Authorized licensed use limited to: Rowan University Libraries. Downloaded on August 29,2022 at 05:33:32 UTC from IEEE Xplore. Restrictions apply.

approach, however, the total run time is only $2T$, i.e., only T seconds per dataset. If this scenario continued for K times, the single-batch version would take $\frac{K(K+1)}{2}T$ seconds, whereas the incremental version would only take KT seconds. For $K = 100$ batches, the difference is $5050T$ vs. $100T$, which is illustrated in Fig.2 comparing the time consumption of VSEARCH and I-VSEARCH to cluster the same amount of increasing bathes of data. VSEARCH is typically around $\mathcal{O}(N^{1.2})$ [7] rather than our assumption above to be $\mathcal{O}(N)$, resulting in more dramatic savings in real world settings.

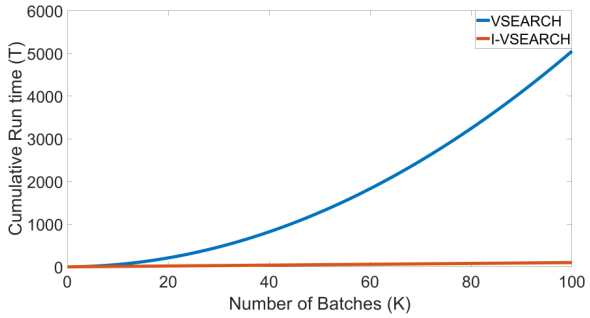

Fig. 2. The cumulative run time of I-VSEARCH is linear, while that of traditional VSEARCH is exponential. T is the nominal time for the algorithm to run once on a dataset of size N sequences.

B. Semi-supervised I-VSEARCH

As indicated above, VSEARCH is a completely unsupervised clustering algorithm. However, the sequences in reference databases always have labels, while most new experimental metagenomic sequences come without labels. In fact, some of the new (unlabeled) sequences may be from previously unknown organisms, for which no taxonomic label yet exists. This setting calls for semi-supervised learning, where there is small amount of labeled data and large amounts of unlabeled data. There is also the additional complexity of some data belonging to new classes that are not yet determined, established or named. To accommodate this very real-life scenario, we took advantage of the availability of small amount of labeled data, and we added a "semi-supervised learning" capability to I-VSEARCH, as described below, in addition to incremental learning capability.

Given that some of the sequences have labels that are known to be correct (as they come from reference datasets), we can use those labels to help label other sequences that fall into the same cluster using majority voting. We then have the following scenarios to consider: in any given cluster (i) there is only one labeled sequence or all labeled sequences are of the same label – in this case, all unlabeled sequences are labeled as the known label; (ii) there are multiple labeled sequences with different labels – in this case, unlabeled sequences are labeled with the most common label i.e., using majority vote; (iii) there are no labeled sequences, in which case the sequences are given a temporary label ID, which is replaced with a true label if or when a known label for that cluster becomes available in a reference database. Fig. 3 illustrates the majority vote labeling process, with clusters 2, 3, and 4 representing the three scenarios listed above, whereas cluster 1 shows a cluster of single sequence.

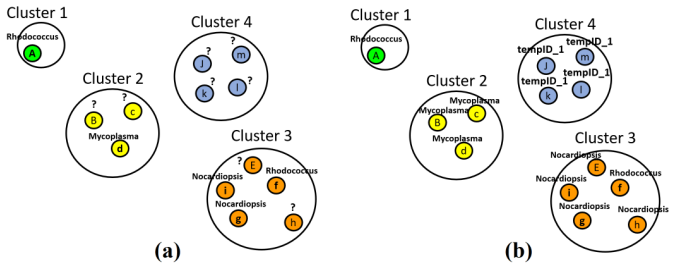

Fig. 3. (a) Clusters generated by VSEARCH, with unlabeled data; (b) Labels provided by semi-supervised version of I-VSEARCH: if there are sequences with different known labels in a cluster, a simple majority vote can be used to label the unlabeled sequences with the most common label in the cluster. It does not matter whether any of the labeled sequences is a seed sequence. Seeds are represented with capitalized letters, sequences with labels are named in bold and unlabeled sequences are represented with a "?".

C. TI-VSEARCH: Threshold Independent VSEARCH

Choosing the value of a free parameter is often a tricky process, one that requires care, as choosing the value incorrectly can lead to poor performance. For VSEARCH the primary free parameter is the similarity threshold. Fig. 4 illustrates the risks of using such a single, global similarity threshold. Here, sample sequences from Species 1 are represented in blue, sample sequences from Species 2 are shown in green, and samples of Species 3 are shown in red. If the similarity threshold is chosen too high (too sensitive), the algorithm provides a good fit to the tight Species 1 and Species 2 clusters, however the larger cluster of Species 3 is broken up into multiple clusters, possibly causing the algorithm to mislabel the subsets of Species 3. On the other hand, if the threshold is chosen too low (not sensitive enough), Species 3 sequences are properly clustered, however, sequences in Species 1 are clustered into the same cluster as Species 2, again resulting in inaccurate clustering and subsequently inaccurate labeling.

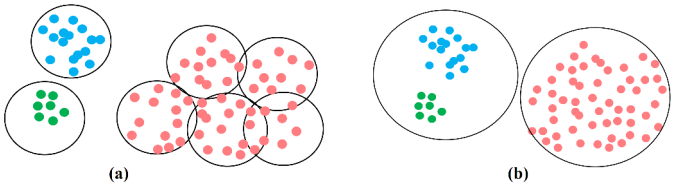

Fig. 4. (a) When the global threshold is too sensitive, a large single cluster is broken up. (b) If the threshold is not sensitive enough, noisy data or unrelated data may be incorrectly clustered.

To address this problem, we developed a threshold independent semi-supervised version of our VSEARCH, referred to as TI-VSEARCH. TI-VSEARCH is a hierarchical algorithm, starting clustering with a less sensitive threshold. After the first pass of a given dataset, TI-VSEARCH looks for *impure* clusters, those whose content include labeled sequences from

Authorized licensed use limited to: Rowan University Libraries. Downloaded on August 29,2022 at 05:33:32 UTC from IEEE Xplore. Restrictions apply.

more than one label (i.e., not all labeled sequences in the cluster are of the same label). TI-VSEARCH then re-clusters all of the data within each such impure cluster at a more sensitive threshold and considers each of the new clusters as sub-clusters. This process forms a tree of clusters and sub-clusters, and is repeated until all clusters within the tree are completely pure. Fig. 5 shows a condensed example of such a tree, which is the result of larger natural clusters being properly fit with a less sensitive threshold while naturally smaller clusters that have been continually sub-divided.

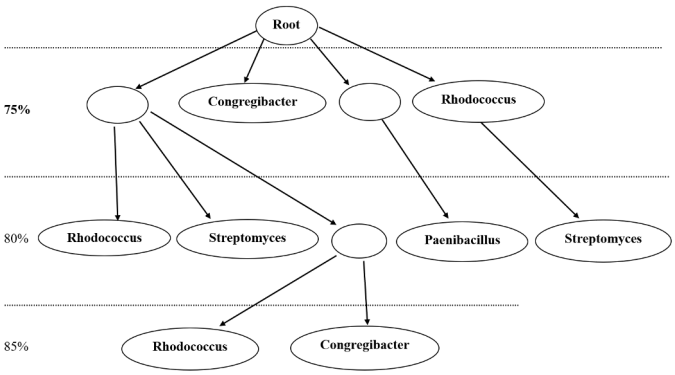

Fig. 5. TI-VSEARCH starts at a low threshold, and continues clustering at higher thresholds until the cluster is pure or it runs out of a list of thresholds.

Fig. 6 shows the flow-chart of the TI-VSEARCH algorithm. Sequences are added to the tree at its root node and propagated down the tree. When a new batch of sequences is added to any of the nodes, standard VSEARCH is run on the new batch of sequences and any sub-clusters of previously labeled data that already exist. All of the new data that joins one of the previous sub-clusters become a new batch of sequences for that sub-cluster, and the process repeats. If a sequence does not join any of the previous sub-clusters, it becomes a seed for a new sub-cluster.

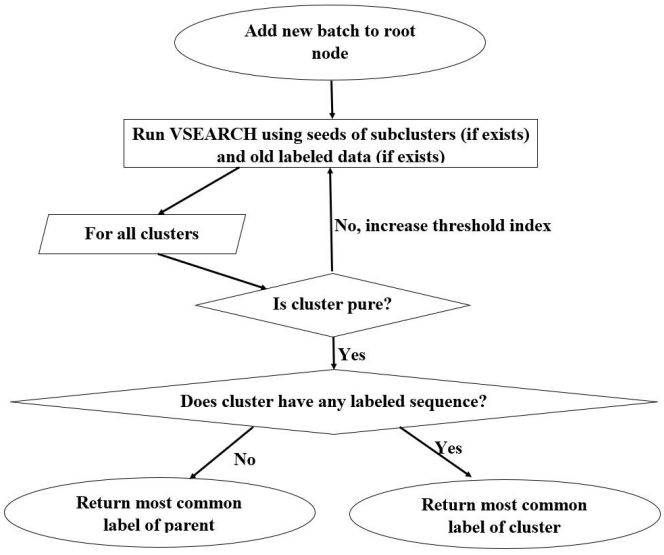

Fig. 6. Flow diagram of the TI-VSEARCH algorithm.

Once new sequences join a sub-cluster, the sub-cluster is purified. If a cluster contains labeled data from only one class, then the cluster is considered pure. The cluster accurately represents only that label, and no other action is required. If there is more than one label in a cluster, then the cluster needs to be subdivided further so that it can be purified. This is done by repeating the above process while increasing the threshold. Clusters that do not have any labeled data cannot be purified.

For prediction of query sequences, data are added to the tree in the same way as labeled data, as described above. In this case, when the sequence reaches a leaf of the tree, the label of that cluster becomes the predicted label. If the cluster has no labeled data, then typically the label of the parent cluster is used, or it can be left unlabeled and the user is informed that the algorithm cannot make a prediction.

The free parameters for TI-VSEARCH include taxonomic depth and a list of thresholds that the user wishes to be evaluated. The threshold list we use in our experiment is the same as typical list of thresholds commonly used [75, 80, 85, 90, 93, 95, 97, 99]. Generally, higher thresholds require more computation time because more sub-clusters are created, which creates more nodes. A larger threshold list typically increases the run time, but not for all nodes, since clustering for a node may stop when purity is reached.

IV. Results and Discussion

A. The RDP Dataset of 16S rRNA Genes

In this proof of concept study, we used the Training Dataset No.14 from the Ribosomal Database Project (RDP) [8]. This dataset consists of 10,679 16S rRNA sequences of Bacteria and Archaea. These sequences are, on average, 1500 base-pairs long, $\pm$ several hundred base-pairs. The dataset contains labels of sequences from the Domain to Genus levels, excluding Kingdom (6 levels of depth). Due to its very slow evolution rate, 16S rRNA gene is suitable for determining phylogenetic relations of species and clustering [9]. 16S rRNA genes are popular biomarkers used by many studies because they are highly conserved, and a natural starting point to incrementalize (due to the millions of sequences from different organisms and thousands of studies that use them).

As a reference database, all sequences in this dataset are in fact labeled. Since VSEARCH is an unsupervised clustering algorithm, the labels are not used in the standard algorithm. In order to evaluate the semi-supervised nature of the I-VSEARCH and TI-VSEARCH algorithms, we have kept the labels of the 25% of the sequences, which then constituted the *training dataset*. The labels for the remaining 75% of the sequences were kept hidden from the algorithms, and these labels were then used as a *test dataset* to compute classification accuracies.

B. VSEARCH and I-VSEARCH

Our first experiment was to verify that there is little or no loss of accuracy with the incremental version of VSEARCH. To do so, we divided the entire RDP dataset into five approximately equal partitions, each representing a batch of

Authorized licensed use limited to: Rowan University Libraries. Downloaded on August 29,2022 at 05:33:32 UTC from IEEE Xplore. Restrictions apply.

data, with sequences in no particular order, that later become available to the algorithm. Recall that 25% of this data have labels available to be used in the semi-supervised stage of the algorithm. We then ran VSEARCH on the entire (full dataset) data in a single batch, as it is normally run. We have then run I-VSEARCH with the 5 batches, sequentially feeding each batch to I-VSEARCH in-order to obtain the final clusters. We provided no access to the data from prior batches as I-VSEARCH ran, except for the seeds as described above. We have varied the similarity threshold from 75 to 97 %, and tracked the labeling accuracy against the known labels (the remaining 75% of the data) at each of genus, family, order, class and phylum taxonomic levels. The results are shown in Fig. 7 for each taxonomic level, comparing the classification accuracy of the standard VSEARCH against the incremental I-VSEARCH. We observe from this figure that there is no loss of accuracy when we run I-VSEARCH incrementally, as compared to VSEARCH, which had the luxury of having access to the entire dataset at once.

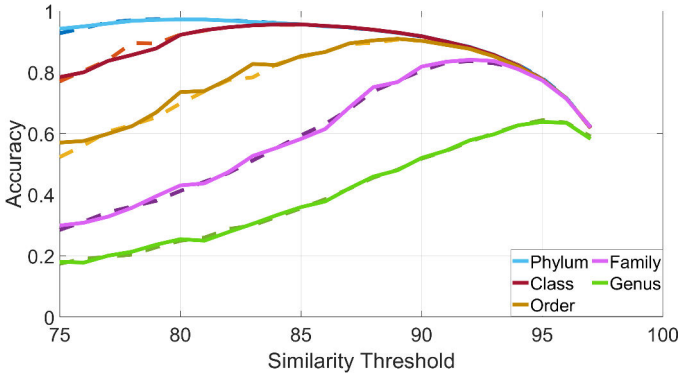

Fig. 7. Classification accuracy comparison between VSEARCH and I-VSEARCH, where majority voting is used to assign labels to unlabeled sequences. Straight lines show VSEARCH performance, whereas dashed lines represent the incremental I-VSEARCH for each taxonomic level.

We also show in Table I the highest accuracies of both methods and the similarity thresholds at which these highest accuracies were observed.

TABLE I
HIGHEST ACCURACIES OF VSEARCH AND I-VSEARCH FOR DIFFERENT TAXONOMIC LEVELS AND CORRESPONDING SIMILARITY THRESHOLDS IN PARENTHESES.

	Phylum	Class	Order	Family	Genus
I-VSEARCH	0.9732 (79)	0.9547 (84)	0.9063 (89)	0.8364 (92)	0.6432 (95)
VSEARCH	0.9719 (81)	0.9561 (85)	0.9090 (89)	0.8405 (92)	0.6375 (95)

The primary advantage of I-VSEARCH is, of course, the ability to process data incrementally and sequentially, but also to provide significant computational savings in doing so. To obtain a quantitative measure on computational savings, we have also compared the two algorithms from a run-time perspective. Since VSEARCH cannot natively run incrementally, we wanted to compare how much of a run-time gain I-VSEARCH provides running incrementally compared to running VSEARCH on the union of all data available at any given time – the only option available to VSEARCH when new data become available. To do so, we ran VSEARCH on the cumulative old+new data, each time a new dataset arrived. On the other hand, Incremental VSEARCH was run only on the new dataset with only cluster seeds from previous run transferred with each new batch. Fig. 8 shows average percentage time saved using I-VSEARCH over rerunning VSEARCH at each of the similarity thresholds and taxonomic levels – on the last (fifth) batch of the experiment. Time saved in Fig. 8 is calculated as

$$TimeSaved = \frac{t_{VSEARCH} - t_{IVSEARCH}}{t_{VSEARCH}} * 100$$

We observe that – regardless of the taxonomic level, I-VSEARCH provides considerable computational savings, particularly at lower similarity thresholds due to number of clusters being significantly fewer. As similarity threshold increases, so does the number of clusters generated and computational cost to make decisions. Recall that Fig. 8 shows time savings at only one step (specifically, at the last of the five batches). We note that increasing the number of batches will only favor I-VSEARCH (as shown in Fig. 2), so in a real world setting of continuously arriving datasets, the run-time saving will accumulate and hence increase over time.

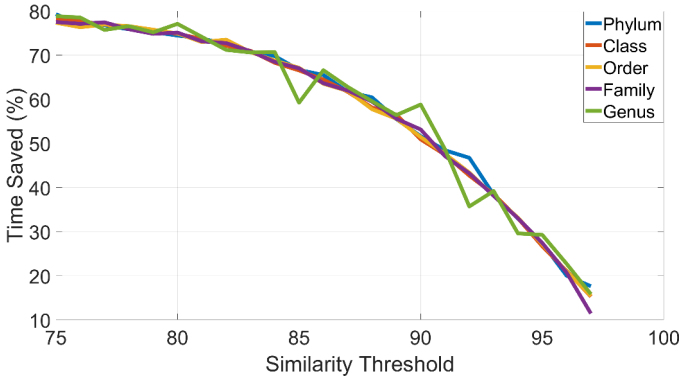

Fig. 8. Percentage time saved when I-VSEARCH used instead of VSEARCH at different thresholds and taxonomic levels.

C. TI-VSEARCH

Fig. 9 compares the classification accuracies of threshold independent VSEARCH (in blue), regular VSEARCH (in yellow) and I-VSEARCH (in purple) across different taxonomic levels. It is important to note that the IV-SEARCH and V-SEARCH results shown in Fig. 9 for each taxonomic level are picked from their peak performances across all values of similarity thresholds (from Table I), whereas the TI-VSEARCH does not use any similarity threshold. We observe in Fig. 9 that all performances of TI-VSEARCH are essentially identical to those of V-SEARCH and IV-SEARCH. This observation means that *the peak performances* of VSEARCH and IV-SEARCH, tuned to the exact optimal threshold, is matched

Authorized licensed use limited to: Rowan University Libraries. Downloaded on August 29,2022 at 05:33:32 UTC from IEEE Xplore. Restrictions apply.

by the performance of TI-VSEARCH, an algorithm that does not depend or use such a threshold. In other words, TI-VSEARCH effectively picks the optimal threshold (in fact, series of thresholds) for the user.

Fig. 9. Performances of threshold independent VSEARCH compared to that of VSEARCH and I-VSEARCH at different taxonomic levels when sequences in unlabeled clusters receive their labels from a parent cluster.

Finally, we also note that in Fig. 9, a sequence that is placed in a cluster with no label receives the label from its parent node. We could choose to leave those sequences unlabeled, and simply warn the user that those sequences are not given a label. Those sequences would simply not be included in the final accuracy computation, which would likely increase the performance. Here, we choose to be more conservative and report the accuracy of the algorithm as obtained by the process described in Section 3.C.

V. Conclusions and Discussion

We have introduced three different modifications to the popular VSEARCH algorithm that is commonly used for metagenomic data clustering. These modifications results in i) IV-SEARCH, an incremental version of VSEARCH that can process different batches of data as they arrive without using previously seen data, ii) a semi-supervised version of IV-SEARCH that can actually label data using small amount of labeled data in training, and iii) Threshold-Independent VSEARCH that relieves the user from the tricky process of choosing the optimal value of the similarity threshold.

I-VSEARCH showed essentially identical classification accuracy as the traditional VSEARCH – despite processing the data incrementally and without having the luxury of seeing all accumulated data – while saving considerable computational time across all taxonomic levels and all values of similarity thresholds (with additional savings at lower similarity thresholds). TI-VSEARCH is just as accurate as VSEARCH / I-VSEARCH, and its built-in threshold optimization makes it more practical to use over other versions. While I-VSEARCH is fastest for the examination of a few threshold and taxonomic level combinations, TI-VSEARCH is more suitable for large optimization problems.

In this effort, we showed a common sequence clustering algorithm can be incrementalized with no loss and considerable computational savings. Since this has been a proof-of-concept study, we evaluated our approach on one particular algorithm (VSEARCH), evaluated on one particular nucleotide 16S rRNA gene dataset. Our future work will include expanding this analysis to other algorithms, platforms, and datasets, including protein datasets. We will also investigate alternative adaptations to TI-VSEARCH and use different levels of purity. By allowing users to set the purity as a free parameter, computation time could be further reduced by allowing a small amount of impurity inside of clusters.

References

[1] J. Handelsman, “Metagenomics: application of genomics to uncultured microorganisms,” Microbiol. Mol. Biol. Rev., vol. 68, pp. 669–685, Dec 2004.

[2] W. Li and A. Godzik, “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.

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

[4] T. Rognes, T. Flouri, B. Nichols, C. Quince, and F. Mah, “Vsearch: a versatile open source tool for metagenomics,” PeerJ, vol. 4, p. e2584, Oct. 2016.

[5] 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. 386, 2009.

[6] B. Buchfink, C. Xie & D. H. Huson, ”Fast and Sensitive Protein Alignment using DIAMOND,” Nature Methods, vol. 12, pp. 59-60, 2015.

[7] A. Rubio-Largo, L. Vanneschi, M. Castelli, and M. A. Vega-Rodríguez. ”Reducing alignment time complexity of ultra-large sets of sequences,” Journal of Computational Biology, vol. 24, no. 11, pp. 1144-1154, 2017.

[8] J.R. Cole, Q. Wang, J. A. Fish, B. Chai, D. M. McGarrell, Y. Sun, C. T. Brown, A. Porras-Alfaro, C. R. Kuske, and J. M. Tiedje. ”Ribosomal Database Project: data and tools for high throughput rRNA analysis,” Nucleic Acids Research, vol. 42, No: D1(Database issue), pp. D633-D642, 2014.

[9] 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, pp. 5088-5090, Nov 1977.

Authorized licensed use limited to: Rowan University Libraries. Downloaded on August 29,2022 at 05:33:32 UTC from IEEE Xplore. Restrictions apply.