Fast-Part: Fast and Accurate Data Partitioning for Biological Sequence Analysis

Shafayat Ahmed, Muhit Islam Emon, Nazifa Ahmed Moumi, Liqing Zhang

Department of Computer Science, Virginia Polytechnic Institute and State University.

*Corresponding author(s). E-mail(s): lqzhang@vt.edu; Contributing authors: shafayatpiyal@vt.edu; muhitemon@vt.edu; moumi@vt.edu;

Abstract

Developing effective machine learning models for classifications of biological sequences depends heavily on the quality of the training and test datasets split. Existing tools are either computationally expensive, unable to maintain the desired level of similarity between the training and test datasets, or unable to retain training: test ratio stratification. Here, we present Fast-Part, a fast and accurate sequence data partitioning tool that ensures strict homology separation between the training and test datasets and the best possible training: test stratification ratio, and at the same time is computationally fast. Evaluation of Fast-Part on multiple protein sequence datasets shows that it performs data partitioning with exceptional speed and maintains strict partitioning compared to the existing tools. Fast-Part can handle massive datasets like CD-HIT[1] and MMseq[2] and maintain strict homology partitioning like GraphPart[3].

Keywords: Data Partitioning, Homology separation, Clustering , sequence partitioning

1 Introduction

Partition of biological sequences into training and testing sets is crucial in developing robust machine-learning models for sequence classification. Random splitting can lead to the overestimation or underestimation of model performance. Among common practices, one of them is stratified k-fold cross-validation. In this approach, the dataset is split into k partitions where sequences are assigned to each partition randomly, and in turn, each of these partitions serves as the test set with the remaining partitions acting as the training set. The overall accuracy is the mean accuracy across all k-folds. Sometimes, instead of multiple partitions, datasets are split into two (train and test) or three sets (train, test, and validation), with the train set typically comprising 60-80% of the data. These methods often fail to maintain sequence homology between the train and test sets, potentially overestimating the model's accuracy[3].

That's why homology-based partitioning comes into play. Homology-based partitioning involves clustering the dataset in such a way that each cluster maintains up to a pre-defined similarity threshold with the other partitions. If the threshold is set at 60%, no sequences from different clusters should have a similarity of more than 60%. To achieve this goal popular tools include CD-HIT[1] and MMseqs[2], both of which can cluster thousands of sequences in a short time. CD-HIT calculates sequence similarity by counting shared k-mers, grouping sequences into clusters based on a specified similarity threshold, with the longest sequence initially acting as the reference. For sequence similarities below 40%, CD-HIT switches to PSI-CD-HIT mode, which utilizes PSI-BLAST for more refined clustering analysis, as described by Hobohm et al.[4]. MMseqs[2] uses a substitution matrix to compare k-mers, identifying high-scoring pairs aligned locally, similar to BLASTP's methodology. Its default clustering mechanism, a greedy set-cover algorithm, selects sequences that connect with most others in the dataset, though it can also employ CD-HIT's strategy for grouping. Both methods group similar sequences into clusters, each with a lead sequence representing all the sequences within that cluster. This reduces computational costs because only the lead sequence is processed instead of every sequence individually. However, this approach does not guarantee that sequences from different clusters have a consistent level of similarity to each other. It ensures similarity within clusters but not necessarily between clusters. The GraphPart[3] addressed this issue by proposing a method that compares pairwise sequence similarity and cluster sequences with strict similarity threshold identity. This method, inspired by Hobohm and colleagues' algorithm from 1992[4], requires calculating all pairwise similarities, a process that requires significant computational complexity making clustering large datasets challenging. Moreover, maintaining a strict similarity threshold across the entire dataset often fails to preserve the stratification ratio for each class. For instance, in classification tasks, significant deviations from the intended train-test ratio for certain courses can lead to underperformance due to insufficient training data or an excessively large test dataset, resulting in skewed model evaluations [5–8]. Tools such as DataSAIL[9] prioritize maintaining the similarity between any two samples across different data splits. Previously, some methods focused on either global [5] or local [6] sequence alignments for reducing sequence homology. Specifically, Graph-Part employed measures like percentage identity and overlap length to conduct global alignment and achieve homology reduction [7, 8, 10].

Challenges such as label imbalance, the presence of numerous labels, and strict similarity thresholds persist and demand careful consideration. When a specific label or class is underrepresented, when the dataset encompasses a significant variety of labels, or when stringent similarity thresholds are enforced, the data partitioning algorithm must be executed with precision. This ensures an effective and fair split of the dataset, crucial for maintaining quality in model training and validation processes. We addressed all these issues with a fast and accurate data partitioning tool that maintains:

1. A strict similarity threshold between sequences from the same label ensures no sequence in the test set has any similarity with train sequences for a given threshold.
2. Speed and efficiency, comparable to CD-HIT/MMseqs, capable of handling extensive data.
3. Close adherence to the stratification ratio for each class, with a minimal margin of error.
4. Retention of as much data as possible through iterative reassignment.
5. A randomized but balanced distribution to ensure the training data encompasses a wide variety and captures more patterns.

Our model demonstrated superior performance over other partitioning tools in achieving these objectives.

2 Materials and Methods

2.1 Datasets

We benchmarked our sequence partitioning tool using four diverse protein sequence datasets, as shown in Table 1. These datasets include the antimicrobial resistance gene classification dataset HMD-ARGDB[11], which contains 18,000 antibiotic resistance genes across 33 different classes, and the DeepLoc dataset[12], which focuses on eukaryotic subcellular localization. We also utilized a Metal Resistance Gene classification dataset[13], along with the large-scale mobileOG-db dataset[14] of mobile orthologous groups—an example where traditional tools like GraphPart often struggle due to their massive size. These four datasets were carefully chosen to evaluate our proposed partitioning algorithm under various conditions, including small and large datasets, single-class and multi-class datasets, and different similarity thresholds. This comprehensive evaluation ensures that our tool is robust and adaptable for diverse biological data analysis tasks.

Dataset	**Number of Labels**	**Size**	**Maximum Length**	**Minimum Length**
HMD-ARGDB	33	18k	1576	34
METAL	1	155k	1895	47
Mobile-OGDB	6	1M	14944	29
DeepLoc	25 (7)	14004	3579	69

Table 1: Total number of sequences, number of labels, max and min length of the sequences for the four datasets.

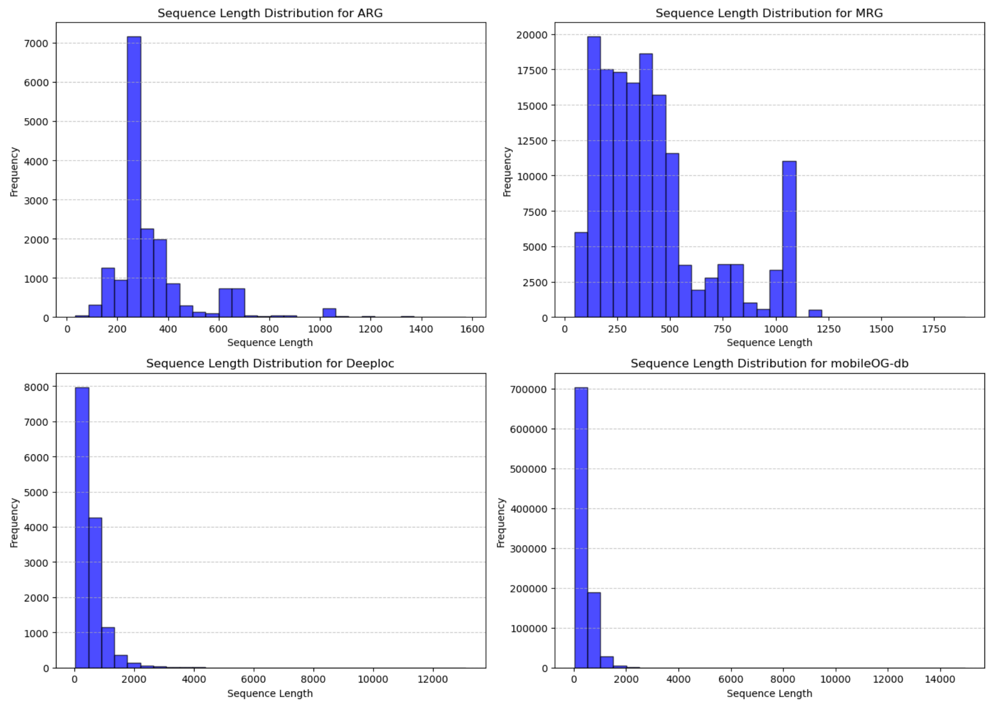


Fig. 1: Sequence length distribution for the four datasets: ARG, MRG, Deeploc, and mobileOG-db.

2.2 Fast-Part Architecture

Figure 2, presents an overview of the FastPart data partitioning tool, designed to meet the specified objectives. The tool accepts a FASTA file, along with parameters such as a similarity threshold, coverage, and a desired stratification ratio as inputs. The output consists of train and test FASTA files, strictly maintaining a similarity threshold between the sets. The tool also ensures that the stratification ratio for each label closely aligns with the user-specified ratio. Additionally, FastPart generates a list of sequences excluded during processing and a text file containing the names of labels that could not be partitioned within the established thresholds for similarity and stratification.

Our pipeline is structured into four critical steps, each essential to the robustness and efficiency of the partitioning process 1. Each step will be detailed to elucidate its role and the methodologies employed.

2.2.1 Division

The initial step involves processing the provided fasta file and splitting it into one or more fasta files, each containing only the sequences associated with a single label.

Algorithm 1 Fast-Part Sequence Clustering

Input:

- *sequences*: List of sequences
- c: Similarity threshold
- f: Stratification ratio
- q: Query coverage threshold

Output: *train_set*, *test_set*

```
function Division(fasta_file):
    return ParseFastaByLabel(fasta_file)
function Clustering(label_fastas, $c$):
    return {label: MMseqsClustering(fasta, $c$) for label, fasta in label_fastas}
function InitialPartition(clusters_by_label, $f$):
    train, test = {}, {}
    for label, clusters in clusters_by_label do
        sorted_clusters = SortBySize(clusters)
        if len(sorted_clusters) == 1 then
            train[label] = sorted_clusters
        else
            test[label] = [sorted_clusters.pop(0)]
            train[label] = [sorted_clusters.pop(-1)]
            DistributeClusters(sorted_clusters, train[label], test[label], f)
    return train, test
function IterativeReassign(train, test, $c$, $q$, $f$):
    while true:
        reassigned = false
        for label in train do
            aligned_seqs = RunDIAMOND(CreateFasta(train[label]), CreateFasta(test[label]), $c$, $q$)
            if aligned_seqs == empty then continue
            for seq in aligned_seqs do
                if MaintainsStratification(train[label], seq, f) then
                    MoveToTrain(train[label], test[label], seq)
                else
                    RemoveFromSmallerSet(train[label], test[label], seq)
                reassigned = true
        if not reassigned then break
    return train, test
function Merging(train, test, $c$, $q$):
    combined_train = MergeClusters(train)
    combined_test = MergeClusters(test)
    aligned_final = RunDIAMOND(combined_train, combined_test, $c$, $q$)
    if aligned_final != empty then
        RemoveFromTrain(combined_train, aligned_final)
    return combined_train, combined_test
function FastPart(fasta_file, $c$, $q$, $f$):
    labels = Division(fasta_file)
    clusters = Clustering(labels, c)
    train, test = InitialPartition(clusters, f)
    train, test = IterativeReassign(train, test, c, q, f)
    return Merging(train, test, c, q)
```

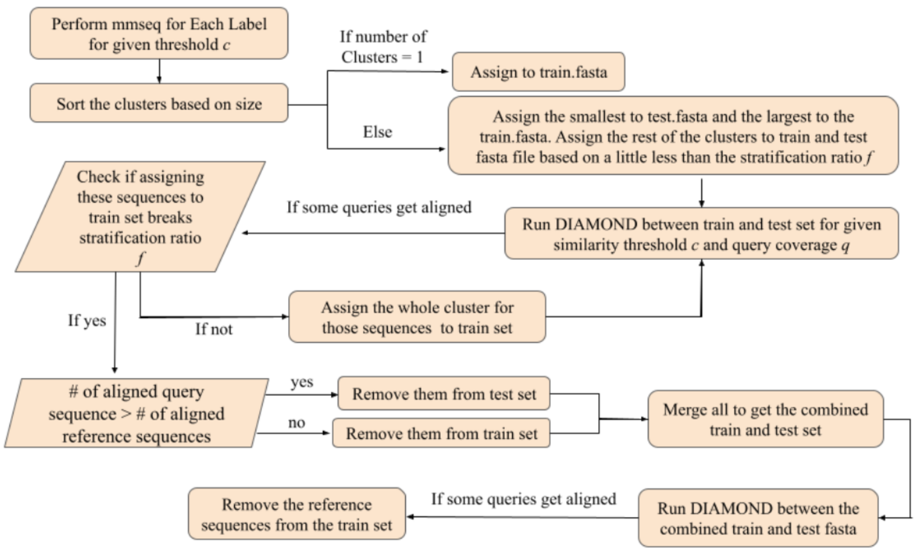

Perform mmseq for Each Label for given threshold c
Sort the clusters based on size
If number of Clusters = 1
Assign to train.fasta
Else
Assign the smallest to test.fasta and the largest to the train.fasta. Assign the rest of the clusters to train and test fasta file based on a little less than the stratification ratio f
Run DIAMOND between train and test set for given similarity threshold c and query coverage q
If some queries get aligned
Check if assigning these sequences to train set breaks stratification ratio f
If yes
If not
Assign the whole cluster for those sequences to train set
# of aligned query sequence > # of aligned reference sequences
yes
no
Remove them from test set
Remove them from train set
Merge all to get the combined train and test set
Run DIAMOND between the combined train and test fasta
If some queries get aligned
Remove the reference sequences from the train set

Fig. 2: A Flowchart of the Fast-Part data partitioning pipeline.

2.2.2 Clustering

The subsequent step involves clustering each fasta file using CDHIT or MMseqs based on the specified similarity threshold. In our Fast-Part tool, we employ MMseqs to expedite the clustering process. After clustering, each fasta file results in one or more clusters. If only a single cluster is generated, it implies that MMseqs cannot partition the fasta file at the defined similarity threshold; thus, this cluster is designated for the training set. If multiple clusters are produced, we assign the largest cluster, along with its representative sequences, to the training set, and the smallest to the test set. The remaining clusters are allocated to the train and test sets randomly but under the specified stratification ratio. Given that CDHIT and MMseqs prioritize maximizing in-cluster similarity, we initially assign clusters slightly below the stratification ratio to facilitate closer alignment with our target partitioning after sequence reassignment.

2.2.3 Iterative Reassignment

Following the initial partitioning based on similarity thresholds and stratification ratios, we assess the partitions for sequence similarity using DIAMOND alignment between the train and test fasta files for each label. If the DIAMOND output indicates similarities between sequences in the train and test sets, we identify these sequences in the test set and their respective clusters for potential reassignment. We then evaluate whether transferring these sequences to the training set would disrupt the stratification ratio for that label. If such a transfer would breach the ratio, we remove the implicated sequences, choosing those from the test set if they are fewer than their counterparts in the training set, and vice versa. If the stratification ratio is unaffected, the sequences are reassigned to the training set. This process is repeated until no sequences in the test set align with those in the training set.

2.2.4 Merging

Upon completion of the iterative reassignment, we compile the train and test fasta files for each label into comprehensive train and test sets. These consolidated sets are structured to prevent any similarity above the defined threshold between sequences of the same label. However, sequences from different labels with homology above the threshold may still be present. To address this, a final DIAMOND alignment is conducted between the combined train and test sets using the established similarity threshold and coverage criteria. Sequences from the training set that align with those in the test set are then removed.

This comprehensive process ensures the preparation of a dataset that adheres to both similarity and stratification ratios, is efficiently processed, and minimizes the removal of sequences, thereby preventing overestimation or underestimation in accuracy reporting. The Fast-Part tool thus ensures optimal utilization of the source dataset for accurate evaluation of machine learning models.

3 Evaluation

To rigorously assess the efficacy of our data partitioning tool, Fast-Part, we conducted an extensive evaluation using various datasets. Our analysis spanned datasets ranging from those with a high number of classification labels to those encompassing minor to massive sequence counts, as well as datasets characterized by a minimal number of labels. We also undertook a detailed examination of runtime, similarity thresholds, and stratification ratios for each dataset.

Additionally, we benchmarked Fast-Part against leading data segmentation tools such as GraphPart. Our comparative studies demonstrated Fast-Part's superior performance, establishing its effectiveness in data partitioning tasks across diverse conditions.

4 Results and Discussion

At first, we experimented with the four datasets for different similarity thresholds and conducted a detailed analysis of the train-test split's performance metrics.

For analyzing the stratification ratio of all the labels of each dataset, we categorize them into five different categories -

1. Balanced (up to 10% variation of the train-test split from the desired stratification ratio)
2. Moderate (up to 20% variation of the train-test split from the desired stratification ratio)
3. Low Sample (Samples with less than 15 sequences). Even a 50-50 separation will be annotated as a Balanced split for them. Otherwise, they will be annotated as a low-sample category.

4. Ambiguous (This is for classes with either a train or test set). This can happen if all the sequences from a label are clustered into a single one for a given similarity threshold or they have high similarity scores with other label samples, thus getting their sequences removed from the train set.
5. Other (This category is for the rest of the labels with a poor stratification ratio, train or test set size have massive deviation (more than 20%) from the desired target segmentation.

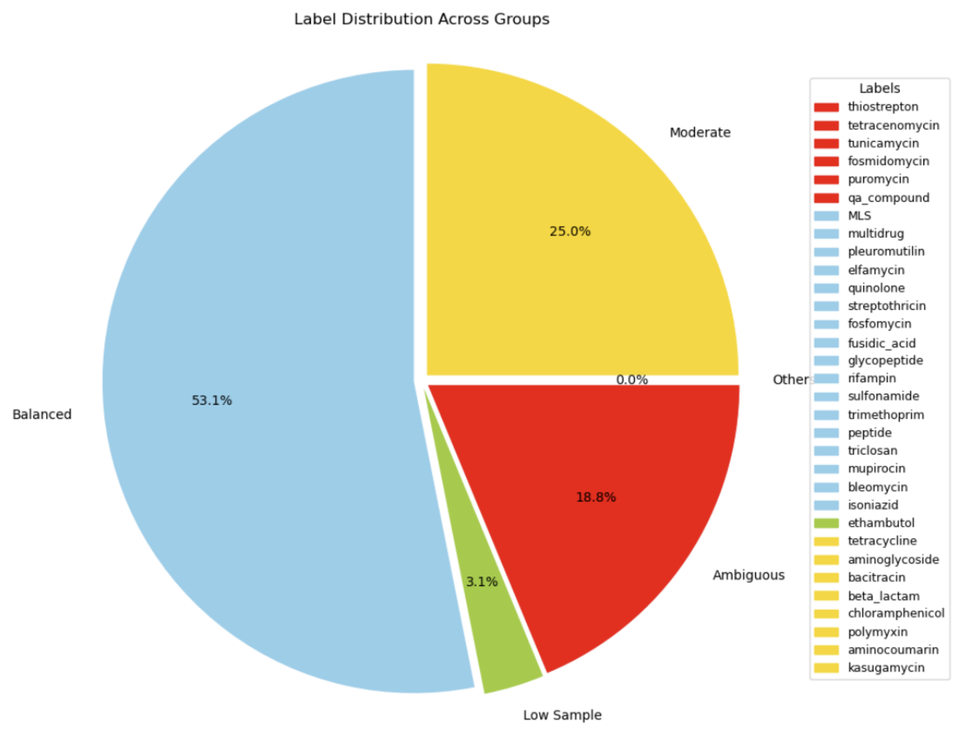


Fig. 3: Stratification ratio for the ARG labels for 40% similarity threshold.

4.1 ARG Dataset

The antimicrobial resistance gene dataset downloaded from HMD-ARGDB has 33 different resistance gene classes. We performed a 40, 60, and 80 percent similarity thresholding on this dataset. The dataset is, however, highly imbalanced as 19 of the ARG classes have, in total, 214 sequences, making it difficult to split it to train and test sets while maintaining the thresholds and stratification ratio. Figure 3, shows the performance of Fast-Part for the most challenging dataset among the four - ARG datasets for a 40% similarity threshold. Still, Fast-Part achieved a 53.1% balanced distribution as well as a 25% moderate split for the dataset. 18.8% (5 of the labels) have either missing train or test samples as it was impossible to split them for a 40% similarity threshold. All of these labels had a meager amount of samples.

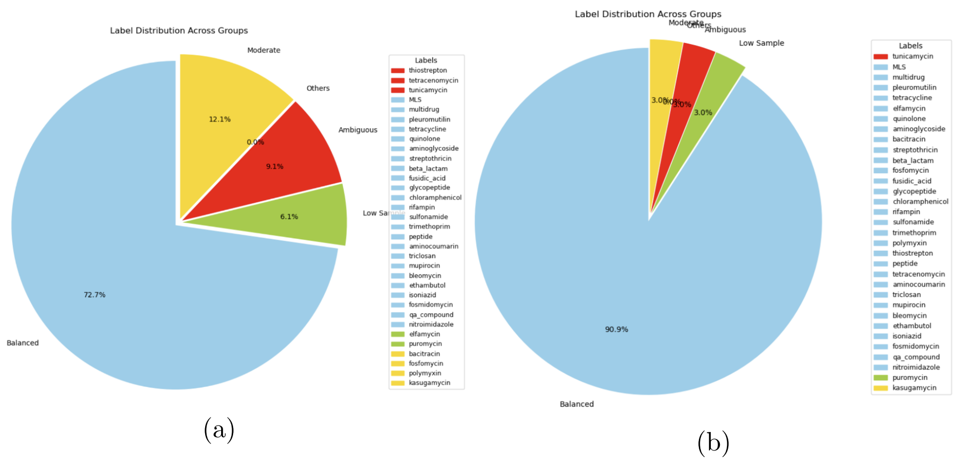


Fig. 4: Stratification report for the ARG dataset's 60% (a) and 80% (b) similarity thresholds.

For 60 and 80 percent similarity thresholds the model performed well considering there were 33 labels for the ARG dataset. Figure 4, shows the distribution of the performance for Fast-Part on 60 and 80 percent similarity thresholds. The model achieves a 72.7 and 90.9 percent balanced distribution of the labels.

4.2 Deeploc

The effectiveness of Fast-Part on the deeploc dataset at different similarity thresholds is showcased in Supplementary Figure 1. Fast-Part maintains consistent performance, ensuring a balanced 80-20 split between training and testing datasets for all labels except the Golgi-apparatus-S label.

4.3 mobileOG-db

Figure 5 demonstrates Fast-Part's consistent performance in partitioning the large mobilOG-db dataset, which includes over a million sequences across six labels. Despite the dataset's size, Fast-Part successfully maintained the stratification ratio, achieving balanced splits for all labels at similarity thresholds of 40%, 60%, and 80% [Supplementary Figure 2].

4.4 MRG

The final dataset, the metal resistance gene dataset, contains only one label. Compared to the other three datasets, this one presented the least challenge, and Fast-Part achieved a near-perfect split (Figure 5), very close to 80-20, at all similarity thresholds (40%, 60%, and 80%).

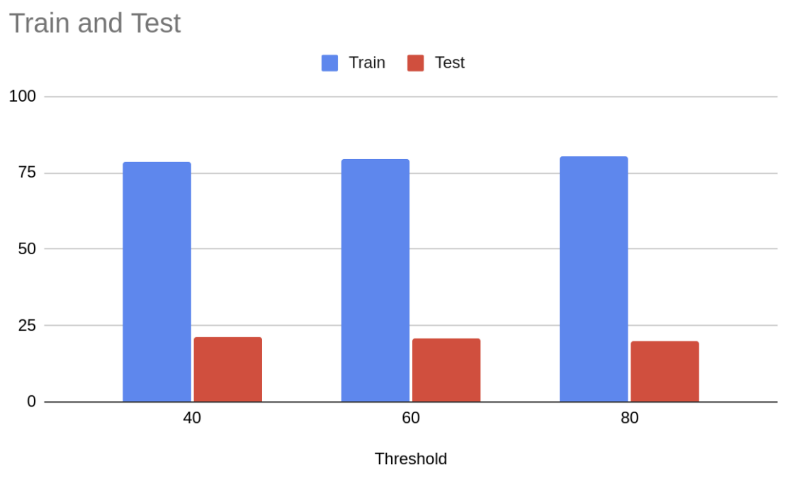


Fig. 5: Stratification report for the MRG dataset at 40,60 and 80 percent similarity threshold.

The initial clustering was done using both CDHIT and MMseqs, We found MMseqs to be a bit faster than the CDHIT and enabled us to retain more sequences Supplementary Table 1. However, we provide both options for the users.

4.5 Removed Sequences

Our model was designed to preserve as many sequences as possible. However, due to the inherent characteristics of the datasets and the requirement to adhere to strict similarity thresholds, Fast-Part had to discard certain portions of the source datasets to ensure balanced distributions. For the deeploc dataset, fewer than 3% of the sequences were omitted even at the 40% similarity threshold. For the metal resistance genes (MRGs), the removal was under 10% across all thresholds. With the extensive mobilOG-db dataset, about 13% of the sequences were eliminated at a 40% threshold, but this figure was significantly reduced to less than 2% at 60% and nearly zero at 80%. For the antibiotic resistance genes (ARGs), the exclusion rate was consistently less than 15% at all thresholds, a notable achievement considering the challenge of maintaining both stratification ratios and similarity thresholds (Figure 6).

4.6 Comparison with Other Tools

We compared our model's efficiency against CDHIT, MMseqs, and GraphPart. Three datasets - ARGs, Deeploc, and the NetGPI dataset were used for data partitioning at a strictly 30 percent similarity threshold for comparison.

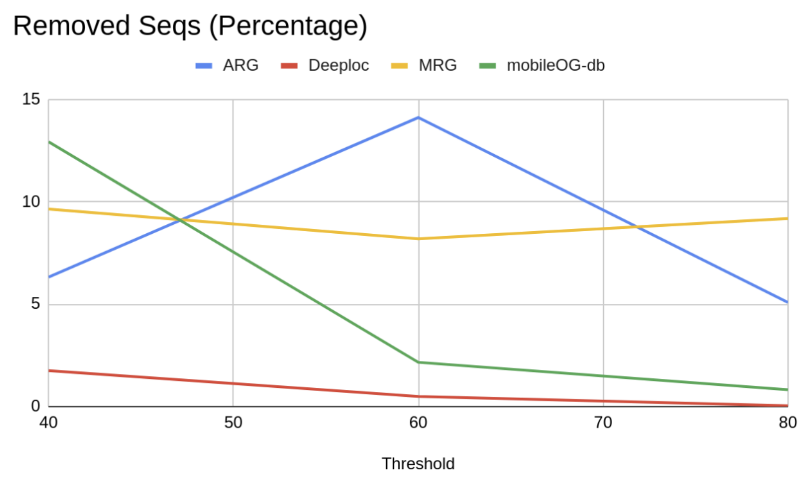


Fig. 6: Percentage of sequences removed for different thresholds for the three datasets ARG, Deeploc, and MRG.

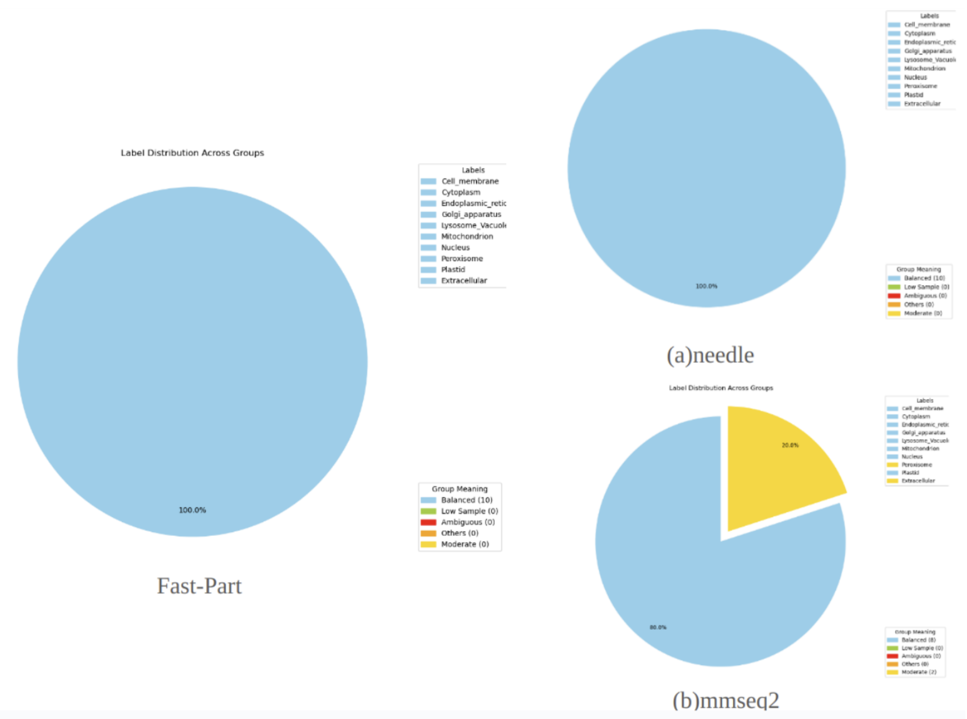
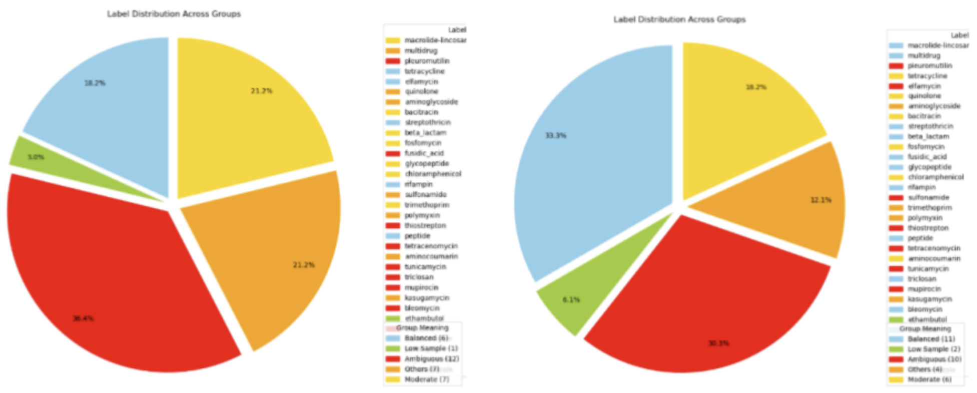


Fig. 7: GraphPart and Fast-Part Accuracy on 30% similarity threshold for the Deeploc Dataset.

GraphPart has two different modes - needle and mmseq2. The needle is recommended but is of quadratic time complexity, whereas mmseq2 is faster but less accurate.

GraphPart and FastPart achieve a complete balanced distribution for the NetGPI dataset. For the Deeploc dataset, the GraphPart needle and FastPart achieve complete balance whereas GraphPart mmseq2 mode achieved 80% balance distribution (Figure 7). However, on both of these datasets, FastPart was at least 2x Faster than the GraphPart needle mode (Table 2). We further tested these models on the challenging ARG dataset and large mobileOG-db datasets. GraphPart needle achieved only an 18 percent balanced distribution whereas mmseq2 got an overall 33 percent balanced distribution for the ARG dataset (Figure 8). Whereas, Fast-Part got a 51% balanced distribution (Figure 9).



(a) needle (b) mmseq2

Fig. 8: Stratification ratio for GraphPart needle and mmseq2 at 30 percent similarity threshold for ARG dataset.

Moreover, GraphPart was exceptionally slow than Fast-Part. Almost 80 times slower for the mmseq2 version and 450 times slower for the needle version for the ARG dataset (Table 2). It was impossible to run the GraphPart needle for the mobileOG-db database due to its huge size.

However, GraphPart was able to retain more sequences compared to the Fast-Part tool for stringent similarity thresholds Table 3.

Furthermore, we analyzed FastPart to check if it maintains a strict similarity threshold between the train and test partition (Figure 10). We saw only GraphPart needle and FastPart were able to consistently retain the threshold similarity strictly for most sequences whereas CD-HIT and MMseqs failed.

For ease of installation, Fast-Part is available both on GitHub and as a Python package via PyPI. You can find the source code and detailed instructions at https://github.com/Shafayat115/Fast-Part. Given a properly formatted FASTA file

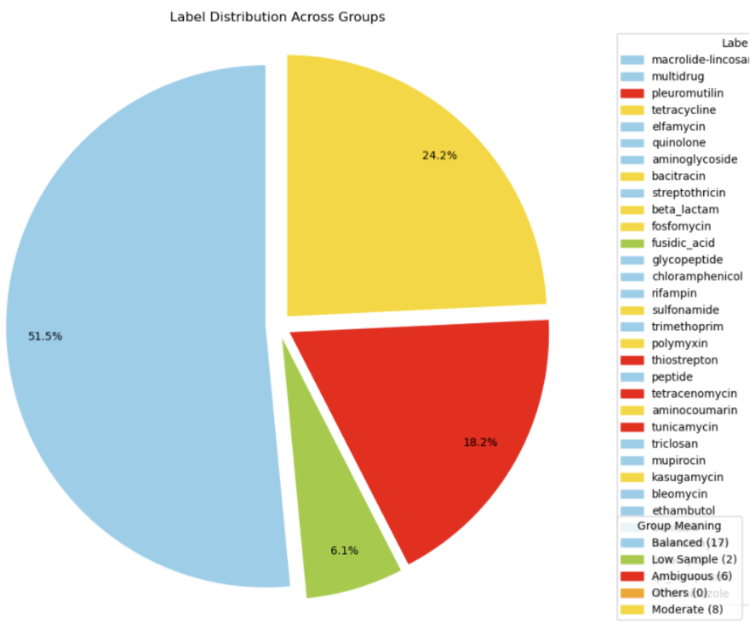


Fig. 9: Stratification ratio for Fast-Part at 30 percent similarity threshold for ARG dataset.

Dataset	**(mmseq2)**	**needle**	**Fast-Part**
Deeploc	816.36	78967	424.62
NetGPI	8.18	469.16	32.6
ARG	8512	39861	90.61

Table 2: Time taken for splitting into train and test set for the 2 GraphPart models and Fast-Part for the three datasets.

Dataset	**Total Seqs**	**mmseq2**	**needle**	**fast-part**
ARG	17282	37	1	823
Deeploc	14004	213	0	519
NetGPI	3618	1	1	146

Table 3: Number of sequences removed for 30 percent similarity threshold train-test split for the three datasets by the two models from GraphPart and Fast-Part.

as input, Fast-Part outputs partitioned training and test files, along with a summary file detailing partition ratios, sequence counts, removed sequences, missing label information, and the configuration settings used for the run. Fast-Part is designed with flexibility in mind, allowing users to input multiple hyperparameters, ensuring it is both powerful and user-friendly. For complete installation instructions and usage details, please refer to the GitHub repository.

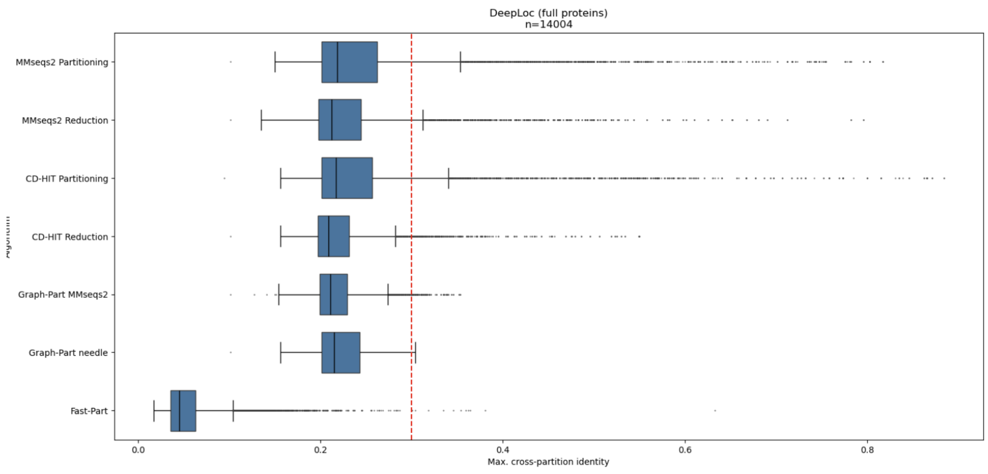


Fig. 10: Similarity distribution between the train and test set for the Deeploc dataset.

5 Conclusion

Data partitioning plays a pivotal role in evaluating sequence classification algorithms, where the accuracy of these methods is heavily contingent upon the integrity of the dataset splits. FastPart offers an advanced solution for partitioning data by ensuring a randomized, yet balanced, distribution of sequences. This tool employs an optimized algorithm to approximate the characteristics of the original dataset while rigorously adhering to predefined similarity thresholds. Additionally, FastPart ensures that the distribution of data adheres closely to the targeted stratification ratios, thus facilitating more reliable and reproducible accuracy assessments in sequence classification endeavors.

Supplementary information.

Acknowledgements.

References

[1] Fu, L., Niu, B., Zhu, Z., Wu, S., Li, W.: Cd-hit: accelerated for clustering the next-generation sequencing data. Bioinformatics **28**(23), 3150–3152 (2012)

[2] Steinegger, M., Söding, J.: Mmseqs2 enables sensitive protein sequence searching for the analysis of massive data sets. Nature biotechnology **35**(11), 1026–1028 (2017)

[3] Teufel, F., Gíslason, M.H., Almagro Armenteros, J.J., Johansen, A.R., Winther, O., Nielsen, H.: Graphpart: homology partitioning for biological sequence analysis. NAR genomics and bioinformatics **5**(4), 088 (2023)

[4] Hobohm, U., Scharf, M., Schneider, R., Sander, C.: Selection of representative protein data sets. Protein Science **1**(3), 409–417 (1992)

[5] Needleman, S.B., Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of molecular biology **48**(3), 443–453 (1970)

[6] Smith, T.F., Waterman, M.S., *et al.*: Identification of common molecular subsequences. Journal of molecular biology **147**(1), 195–197 (1981)

[7] Lund, O., Frimand, K., Gorodkin, J., Bohr, H., Bohr, J., Hansen, J., Brunak, S.: Protein distance constraints predicted by neural networks and probability density functions. Protein Engineering **10**(11), 1241–1248 (1997)

[8] Rost, B.: Twilight zone of protein sequence alignments. Protein engineering **12**(2), 85–94 (1999)

[9] Joeres, R., Blumenthal, D.B., Kalinina, O.V.: Datasail: Data splitting against information leakage. bioRxiv, 2023–11 (2023)

[10] Sander, C., Schneider, R.: Database of homology-derived protein structures and the structural meaning of sequence alignment. Proteins: Structure, Function, and Bioinformatics **9**(1), 56–68 (1991)

[11] Li, Y., Xu, Z., Han, W., Cao, H., Umarov, R., Yan, A., Fan, M., Chen, H., Duarte, C.M., Li, L., *et al.*: Hmd-arg: hierarchical multi-task deep learning for annotating antibiotic resistance genes. Microbiome **9**, 1–12 (2021)

[12] Almagro Armenteros, J.J., Sønderby, C.K., Sønderby, S.K., Nielsen, H., Winther, O.: Deeploc: prediction of protein subcellular localization using deep learning. Bioinformatics **33**(21), 3387–3395 (2017)

[13] Pal, C., Bengtsson-Palme, J., Rensing, C., Kristiansson, E., Larsson, D.J.: Bacmet: antibacterial biocide and metal resistance genes database. Nucleic acids research **42**(D1), 737–743 (2014)

[14] Brown, C.L., Mullet, J., Hindi, F., Stoll, J.E., Gupta, S., Choi, M., Keenum, I., Vikesland, P., Pruden, A., Zhang, L.: mobileog-db: a manually curated database of protein families mediating the life cycle of bacterial mobile genetic elements. Applied and environmental microbiology **88**(18), 00991–22 (2022)