When does additional information improve accuracy of RNA secondary structure prediction?

Logan Rose, Luis Sanchez Giraldo, Duc Nguyen, Matthew Wheeler, David Murrugarra

October 2024

Abstract

The secondary structure of an RNA sequence plays an important role in determining its function, and accurate prediction of the structure is still a major goal in computational biology. Improvements in the prediction accuracy of the secondary structure can be achieved via auxiliary information. In this paper, we study features based on suboptimal formations competing with the minimum-free energy formation and investigate their role in determining the improvement of accuracy via auxiliary information, which we call directability. Here, we introduce a similarity measure among competing substructures called profiles. Then, we present an n-dimensional representation of the profiles which allows the use of topological data analysis (i.e., persistence landscapes) to obtain different metrics that represent topological features. Then, we built random forest classifiers using these novel features. We show how the similarity feature is more important for classifiers trained on sequences with similar structures while the topological features are more important for classifiers trained on sequences with dissimilar structures. We perform extensive testing on two sets of RNA sequences where we studied the sensitivity of the classification accuracy and their feature importance.

1 Introduction

An RNA chain folds back to form higher dimensional structures such as patterns of base pairings forming helices and single stranded loops. The two-dimensional formation is called the *secondary structure of an RNA sequence*. Figure 1 shows the secondary structure of *tRNA* which was generated using the software VARNA [1]. The secondary structure plays an important role in determining the function of an RNA sequence, where a mutated structure might be involved in several diseases [2, 3].

One of the most common tools for computational prediction relies on the nearest neighbor thermodynamic model (NNTM) [4], which is based on energy minimization using dynamic programming. There are

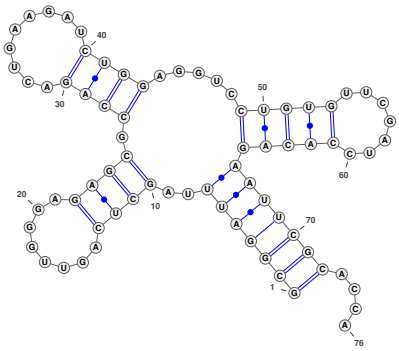

Figure 1: *tRNA* secondary structure which was generated using VARNA [1].

several implementations of NNTM including *RNAstructure* [5], the *ViennaRNA* Package [6], *UNAFold* [7], and *GTfold* [8]. Although very useful, the physics-rooted NNTM approach is prone to accuracy errors in its predictions [9, 10]. Accuracy can be improved by incorporating additional information, such as high-throughput chemical probing data that further constrain the thermodynamic model [11, 12, 13]. However, providing such additional information directly requires costly procedures [11]. Alternative methods that predict such auxiliary information using machine learning (ML) and deep learning (DL) have been proposed [14, 15, 16]. In recent years, data-driven methods for RNA structure prediction based on machine learning (particularly deep learning) have also been proposed [3, 17, 16, 18]. These methods are based on known sequence to secondary structure maps to learn a model to predict the secondary structure of unseen sequences.

The newer methods based on ML, especially deep learning, for RNA structure prediction seem to provide higher accuracies but give little insight about the mechanisms for folding. These methods may be able to learn nuances missed by the NNTM, but rely heavily on training data. Therefore, the imbalance of RNA families in training data sets and the fact that different RNA families have different patterns can create prediction issues, such as overfitting from deep learning methods, where a method performs well in one RNA family and underperforms in others [15, 16]. Addressing these challenges will require going beyond the single minimal free energy prediction, such as studying the suboptimal structure formations to understand the competing alternatives. Boltzmann sampling of secondary structures [19] allows the study of the competing substructural alternatives. RNA structure profiling [2] is a tool based on graph theory techniques that allows the representation of the competing substructures as a partially ordered set that highlights the competing substructures in a Boltzmann sample. Many of these tools have not been fully exploited yet for predicting the secondary structure of RNA.

In this paper, we investigate topological features from the graph of competing substructures that are relevant for determining whether a sequence can significantly improve its predicted accuracy using synthetic auxiliary data. To this end, we construct classifiers that take these features as input to predict 1) directability (i.e., improvements of accuracy via auxiliary information) and 2) well-prediction (that is, if the accuracy of the strucuter prediction will be above certain threshold). The rest of the paper is structured as follows. In the Background section, we describe the RNA structure profiling approach. In the Methods section, we introduce our similarity measure among profiles and describe an n-dimensional embedding for these profiles to perform topological data analysis. In the Results section, we report the accuracy and feature importance of different classifiers for directability and well-prediction. Finally, we describe the main takeaways of the paper in the Conclusions section.

2 Background

2.1 RNA Structure Profile graphs

For an RNA sequence, we obtain features based on suboptimal formations competing with the minimum-free energy (MFE) formation using RNA structure profiling. RNA structure profiling [2] is a tool for understanding competing structural alternatives based on Boltzmann samplings [19]. RNA profiling provides a summary graph as a poset (i.e., a partially ordered set) that highlights the competing substructures from a Boltzmann sample. Figure 2 shows a profile graph for *tRNA*. The table in the top left provides the information of the selected profiles (those with frequency above a certain threshold). Boxed vertices indicate selected profiles, while dashed ovals indicate the intersection of two profiles. Each node is labeled with the profile in parenthetic notation along with its specific and general frequencies written as a ratio.

The profiling method uses the triplet notation of a helix. For instance, the first helix $h = (1, 72, 8)$ in the table of Figure 2 has 8 consecutive base pairs $\{(1, 72), (2, 71), (3, 70), (4, 69), (5, 68), (6, 67), (7, 66), (8, 65)\}$. In general, consecutive base pairs $\{(i, j), \ldots, (i + k - 1, j - k + 1)\}$ form a helix $h = (i, j, k)$. A helix (i, j, k) is maximal if $(i - 1, j + 1)$ and $(i + k, j - k)$ would be non-canonical base pairs or violate $j - i - 2k \geq 2$. Two helices x and y are related if their base pairs are a subset of the same maximal helix h. The equivalence class under this relation is called a helix class, denoted by $[h]$. Given a Boltzmann sample $\mathcal{S}$, the frequency of $[h]$

Helix	Triplet	Frequency
1	1 72 8	988
2	12 35 6	671
3	19 28 2	541
4	45 60 4	540
5	39 66 4	523
6	49 65 5	343
7	27 43 5	256
8	10 25 4	220
9	11 37 2	189
10	32 37 1	149
11	16 30 4	128
12	37 48 4	116

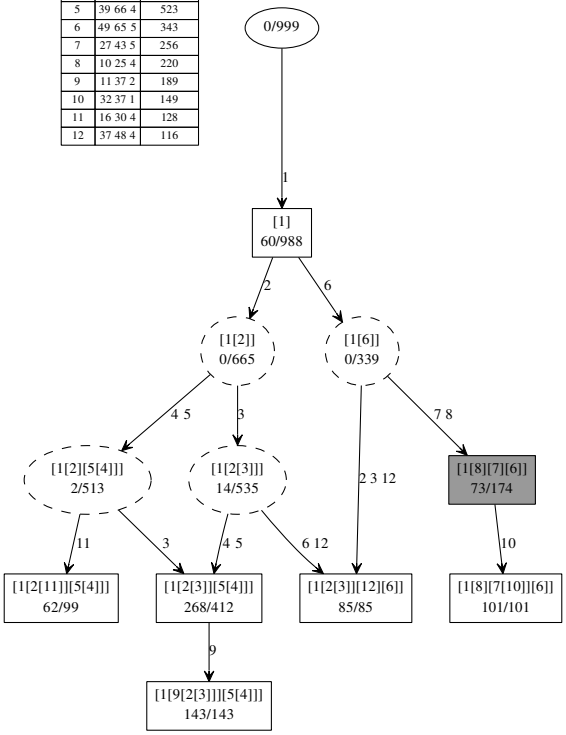

Figure 2: Profile graph for *tRNA*. The table in the top left provides the information of the selected profiles. The shaded profile is the native profile corresponding to the correct structure. Figure generated using RNA Structure Profiling [2].

is $\text{freq}([h]) = \{s \in \mathcal{S} : [h] \cap s \neq \emptyset\}/|\mathcal{S}|$. Profiling groups together similar substructures using helix classes.

The profile p_s of a structure s is the set of its corresponding helix classes that can be presented in parenthetic notation. For instance, the shaded profile in Figure 2 is composed of four helix classes and is represented by [1[8][7][6]]. The set of base pairings (in triplet notation) of the composing helix classes are provided in the table at the top left of Fig. 2.

2.2 Topological Data Analysis

With the continual increase in size and complexity of data sets, building analytical tools for understanding this complexity is important. Many of these complex data sets admit interesting topological and geometric features. For low-dimensional data, this can many times be inferred by inspection, but for high-dimensional data, more advanced topological tools are required. Topological data analysis focuses on the development and implementation of computational tools built on ideas from algebraic topology to identify and quantify these features. One particularly versatile tool is that of Persistent homology which ascribes a filtration of simplicial complexes $K_\bullet$ to a given data set,

$$\emptyset = K_0 \subset K_1 \subset \cdots K_n = K.$$

There are a number of ways to construct these filtrations from point cloud data, and in what follows we will use the Vietoris-Rips filtration [20]. Given a filtration, one then computes the homology at each step of the filtration which induces a morphism of groups for each inclusion map of the filtration. The resulting persistence module is a sequence of group morphisms

$$0 = H(K_0) \to H(K_1) \to \cdots \to H(K_n).$$

Any topological feature in this persistence module is born at some filtration level b and then dies at a subsequent filtration level d. Usually this topological information can be encoded as a persistence diagram, a collection of points $D = \bigoplus I_i \subset \{(x, y) \in \mathbb{R}^2 | \, x < y\}$, representing these topological features where each tuple $I_i = (b_i, d_i)$ corresponds to the birth and death of a feature [21]. However in order to perform statistical analyses, a more appropriate topological summary for our purposes is the persistence landscape [22, 23]. Given a persistence diagram D, the persistence landscape can be defined as follows. Consider the functions

$$f_{(b,d)}(t) = \max(0, \min(a + t, b - t)).$$

The persistence landscape is then given by the sequence of functions $(\lambda_1, \lambda_2, \ldots)$ where

$$\lambda_k(t) = \text{kmax}\left\{f_{(b,d)}(t)\right\}_{(b,d)\in D}.$$

The space of persistence landscapes forms a Hilbert space with norm given by

$$\langle \lambda, \rho \rangle = \sum_{k=1}^{\infty} \int \lambda_k(t)\rho_k(t)dt.$$

Persistence landscapes have a number of nice properties that make them ideal for statistical analysis. First, they are stable with respect to the original point cloud, that is, small perturbations of the data result in only small perturbations of the corresponding landscapes. Being a Hilbert space, persistence landscapes also come with operations of addition and scalar multiplication allowing one to calculate averages, standard deviations, and variances of persistence landscapes as opposed to persistence diagrams.

2.3 Embedding profile graphs into an n-dimensional hypercube

In order to apply topological data analysis tools to extract features based on the the properties of the profile graph, we map the vertices of the profile graph G to points of an n-dimensional hypercube D. That is, we map each vertex of G to a point in D as follows. First, we list all helix classes $h_1, \ldots, h_n$ in the Boltzmann

sample. Then, we use the ratio of the specific frequency of the profiles to the frequency of the composing helix classes to represent the profiles as points in $\mathbb{R}^n$. More specifically, we use the binary random variable $X^p_{h_i}$ that indicates the presence of the helix class h_i in profile p. Then, we define the coordinates of the corresponding point as follows. Let

$$C(X^p_{h_i}) = \begin{cases} \text{freq}(p)/\,\text{freq}(h_i) & \text{if } X_{h_i} = 1 \\ 0 & \text{if } X_{h_i} = 0. \end{cases}$$

Finally, each profile with p is mapped to the point $(C(X^p_{h_1}), \ldots, C(X^p_{h_n})) \in \mathbb{R}^n$. Figure 3 shows the profile graph for *Adenine* and its hypercube representation. This profile graph has only 1 node and the corresponding point is shown in red in Figure 3. We use this hypercube representation to get novel topological features for the classifier of RNA sequence directability.

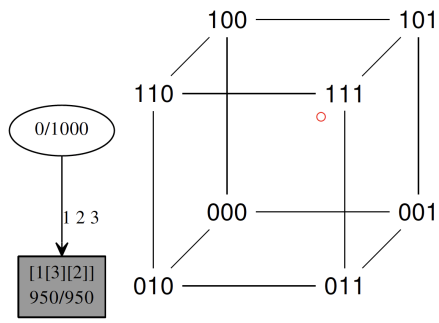

Figure 3: Profile graph of a boltzmann sample for *Adenine* and its hypercube representation. The shaded profile is the native profile. The red dot represents the point: $\left(\frac{950}{1000}, \frac{950}{972}, \frac{950}{950}\right)$.

3 Methods

Here we define the similarity measure among profiles and the norms of persistence landscapes that will represent topological features of the profile graph. Then, we will use these metrics to extract relevant features for our directability and well-prediction classifiers.

3.1 Similarity and variance measures for profiles

For RNA structure prediction, the standard measures for assessing the accuracy of the predictions are the positive predictive value (PPV), $\text{PPV}(P, P^*) = \frac{TP}{TP+FP}$, and the sensitivity (SEN), $\text{SEN}(P, P^*) = \frac{TP}{TP+FN}$, where

- TP: true positives. Base pairings that occur in both the native and the predicted structure.
- FP: false positives. Base pairings that occur in the predicted structure but not the native.
- FN: false negatives. Base pairings that occur in the native structure but not the predicted.
- PPV: is the fraction of true positives in the predicted structure.
- SEN: is the fraction of true positives in the native structure.

While SEN and PPV have been primarily used as standard measures for structures, they can also be applied for RNA profiles in the following way. Given a profile P in a Boltzmann sample, we take the MFE profile P^* as a reference and compare P to P^* using,

$$\mathrm{PPV}(P, P^*) = \frac{TP}{TP + FP} \text{ and } \mathrm{SEN}(P, P^*) = \frac{TP}{TP + FN}$$

where

$$TP = \sum_{HC} \mathrm{L}(HC), \text{ where } \mathrm{L}(HC) \text{ is the length of a true positive helix class.}$$

The summation is over all true positive helix classes. FP and FN are defined in a similar way. $\mathrm{L}(HC)$ could also be defined as the average length of helices in the helix class. We take the MFE profile as a reference because we want to measure how centered is the sample in the MFE structure.

For each profile P_i, we compute $\mathrm{PPV}(P_i, P^*)$ and $\mathrm{SEN}(P_i, P^*)$. Then, we define the similarity of P_i and P^* as the arithmetic mean of $\mathrm{PPV}(P_i, P^*)$ and $\mathrm{SEN}(P_i, P^*)$:

$$\mathrm{SIM}(P_i, P^*) = \frac{\mathrm{PPV}(P_i, P^*) + \mathrm{SEN}(P_i, P^*)}{2}$$

Since each profile comes with its frequency, we can compute the weighted average of the $\mathrm{SIM}(P_i, P^*)$'s,

$$\mathrm{SIM} = \frac{\sum_{P_i} \mathrm{freq}(P_i)\, \mathrm{SIM}(P_i, P^*)}{\sum_{P_i} \mathrm{freq}(P_i)}$$

and the weighted sample SIM variance

$$\sigma^2 = \frac{\sum_{P_i} \mathrm{freq}(P_i)(\mathrm{SIM}(P_i, P^*) - \mathrm{SIM})^2}{\sum_{P_i} \mathrm{freq}(P_i)}.$$

3.2 Persistence Landscapes

To detect topological features of the structure profile graph, we used persistence landscapes as a topological summary. For each profile graph, the corresponding embedding into the n-dimensional hypercube maps each profile p as

$$p \mapsto C(X^p) = (C(X^p_{h_1}), \ldots, C(X^p_{h_n})) \in \mathbb{R}^n$$

and thus the profile graph can be viewed as a point cloud in $\mathbb{R}^n$.

To compute persistent homology, we constructed the Vietoris-Rips complex on this point cloud. For each $r > 0$, one associates the collection of simplices $[p_1, \ldots, p_n]$ satisfying $d(C(X^{p_i}), C(X^{p_j})) < r$ for all p_i, p_j in the simplex, where d here is the usual Euclidean metric. From there one computes homology of varying degree and subsequently the persistence landscapes.

All computations for calculating persistent homology and persistence landscapes were done using the tda-tools package in R. The landscapes were discretized by using an interval of 0.01 transforming each landscape into a high-dimensional vector. From there we computed the Euclidean norms of these landscapes. By taking homology of each degree k, we have a corresponding norm which we denote as the k-norm. We used the 0-norm and the 1-norm as our final summaries. The 0-norm describes the persistence of the connected components of the profile graph where the connected components correspond to each profile of the profile graph. The 1-norm then corresponds to the persistence of all 1-dimensional holes in the profile graph.

3.3 Synthetic auxiliary data

Experimentally derived auxiliary data such as Selective 2'-hydroxyl acylation analyzed by primer extension (SHAPE) has been used to improve accuracy. In this work, we use synthetically generated auxiliary data using the approach described in [15]. For an RNA sequence, the method in [15] produces a sequence p of

the same length as the original RNA sequence, where p_i is the predicted probability that the nucleotide in position i is paired. Using these predictions, one can convert each predicted probability p_i to a SHAPE value to be associated with nucleotide i. Then, the following piecewise linear function is used to generate new synthetic auxiliary data:

$$f(i) = \begin{cases} 2(0.3603 - b)p_i + b, & \text{if } \ 0 \leq p_i \leq 0.5 \\ 2(a - 0.3603)(p_i - 1) + a, & \text{if } \ 0.5 < p_i \leq 1 \end{cases} \tag{1}$$

We list the main properties of this function below:

- This function has range $[a, b]$,
- $f(i) = a$ if $p_i = 1$,
- $f(i) = b$ if $p_i = 0$, and
- $f(i) = 0.3603$ if $p_i = 0.5$.
- To determine the values of a and b, we considered experimentally collected SHAPE data from two E. coli sequences: one 16S sequence and one 23S sequence from [12].
- We took the mean SHAPE value among both paired nucleotides and unpaired nucleotides; these values are 0.214 and 0.6624, respectively. Our experimental results used $a = 0.214$ and $b = 0.6624$.

For this paper, since our objective is to evaluate the role of the features, we assume to know the state of the nucleotide (as being paired or unpaired), and we generate our synthetic data with this information instead of using a predictor of the pairings.

3.4 Features

We considered the following features for the classifiers. Here, we provide the abbreviations that we will use in the figures in the Results Section:

1. Sim: the similarity before SHAPE.
2. Var: the variance before SHAPE.
3. SHS: the similarity after SHAPE.
4. VSH: the variance after SHAPE.
5. N0: the 0-norm before SHAPE.
6. N0SH: the 0-norm after SHAPE.
7. N1: the 1-norm before SHAPE.
8. N1SH: the 1-norm after SHAPE.

3.5 Test sequences

We have trained and tested several classifiers using known data sets of RNA sequences that have been used for other benchmarking purposes [24, 3]. We have used TrainSetA, TestSetA, TrainSetB, and TestSetB, which are available in the supplementary material of [3]. These data sets were originally established by Rivas et al. in [24]. As noted in [3], TrainSetB and TestSetB contain dissimilar structures, while TrainSetA and TestSetA have similar structures.

The MFE accuracies before and after synthetic SHAPE for sets TrainSetA and TrainSetB are shown in Figures 4 and 5. We obtained these accuracies using RNAStructure [5].

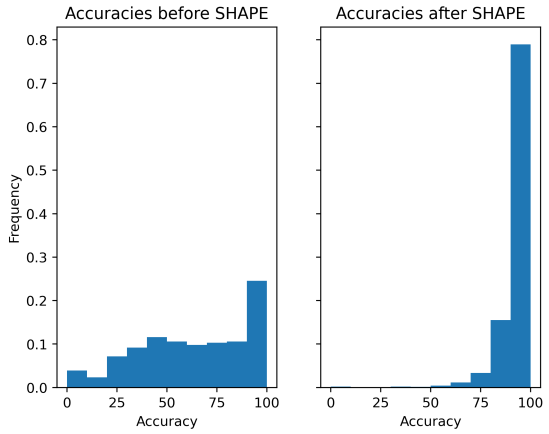

Figure 4: Accuracies before and after synthetic SHAPE for TrainSetA.

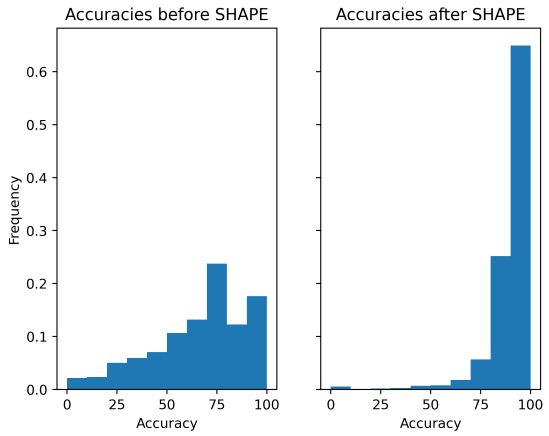

Figure 5: Accuracies before and after synthetic SHAPE for TrainSetB.

4 Results

In this section, we report the accuracy and feature importance of the different classifiers for well-prediction and directability. Our classifiers are based on the Random Forest Classification algorithm, which we selected as an appropriate tool for this kind of task.

4.1 Classifying Well-Prediction

We built classifiers that take an RNA sequence as an input and then predict whether the secondary structure of the sequence is well-predicted (WP) or not (NWP) by the NNTM model, see Figures 6-7. We considered an RNA sequence as being WP if its NNTM accuracy (i.e., the MFE accuracy) is higher than an accuracy threshold. Otherwise, the sequence is classified as NWP. We varied the accuracy thresholds from 50% to 60%, these are given in the y-axis of Figures 6-7. Our classifiers were created in Python using the RandomForestClassifier implementation from SciKitLearn. Each RandomForestClassifier object contains the following hyperparameters: **n_estimators** (the number of trees in the random forest), **max_depth** (the maximum depth of a tree), and **random_state** (determines the random number generator used). Unless

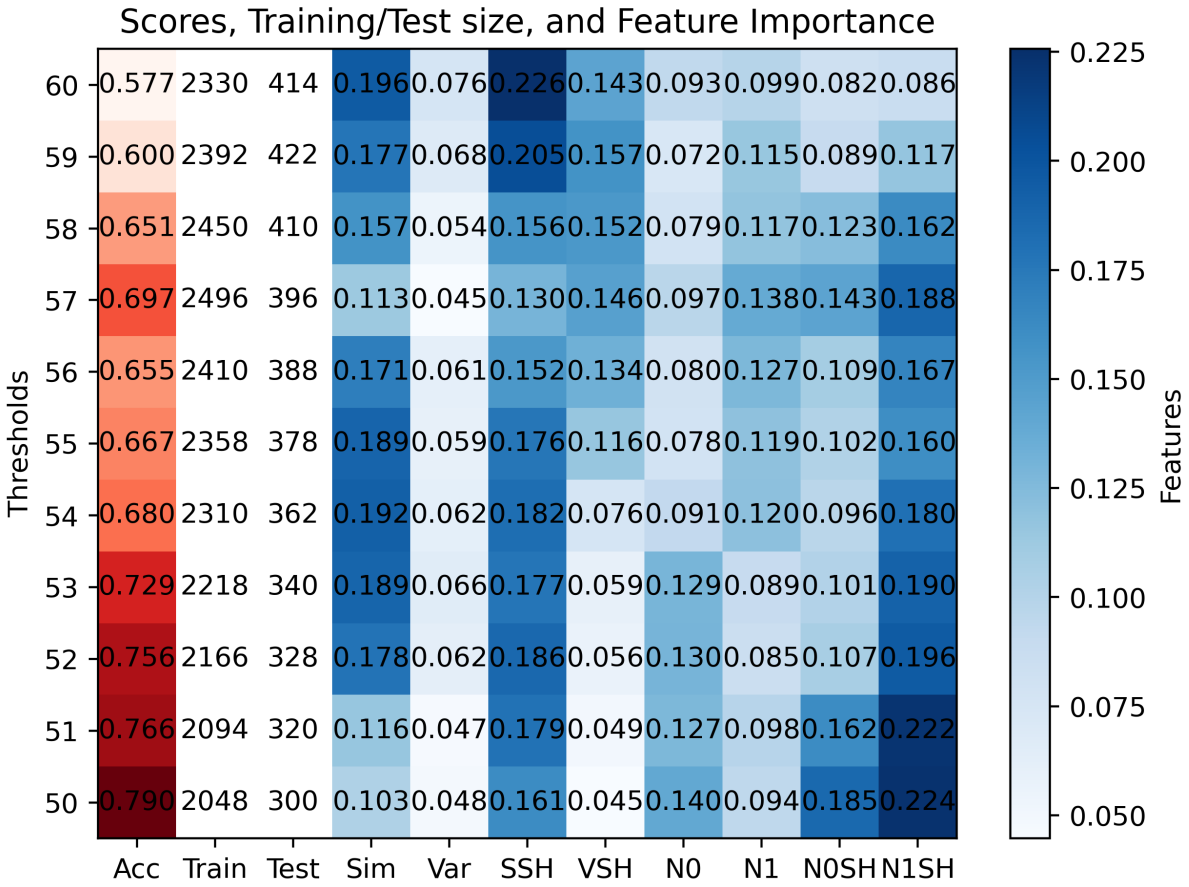

Figure 6: Classifying well-prediction on TestSetA. The classifiers were trained using sequences from TrainSetA and then tested using sequences from TestSetA.

stated otherwise, we set *n_estimators*=1000, max_depth=2, and random_st=0. Classifiers were constructed using one of two training sets (TrainSetA and TrainSetB). Training sets were balanced to contain an equal number of WP and NWP sequences. We tested classifiers using data from one of two test sets (TestSetA and TestSetB) and obtained a **score** from Python, which is the percentage of sequences in the test set that were correctly identified as WP or NWP.

From Figure 6, which corresponds to classifiers trained in sequences in TrainSetA and tested in sequences in TestSetA, we see that for the task of well-prediction, the most important features are the similarities before SHAPE (Sim), the similarities after SHAPE (SSH), and the 1-nom after SHAPE (N1SH). And, in Figure 7, which corresponds to classifiers trained in sequences in TrainSetB and tested in sequences in TestSetB, we see that for the task of well-prediction, the most important features are the similarities after SHAPE (SSH) and the 0-norm after SHAPE (N0SH). The reason for the difference in feature importance between the classifiers trained in TrainSetA and the classifiers trained in TrainSetB is likely due to the composition of these two training sets of sequences. As noted in [3], TrainSetB and TestSetB contain dissimilar structures, while TrainSetA and TestSetA have similar structures. Thus, we expect the similarity feature (Sim) to be more important for classifiers trained on TrainSetA than for classifiers trained on TrainSetB. We also note that the 0-norm after SHAPE feature (N0SH) is more important for classifiers trained in TrainSetB than for classifiers trained in TrainSetA, indicating that the mechanisms for improving accuracy through

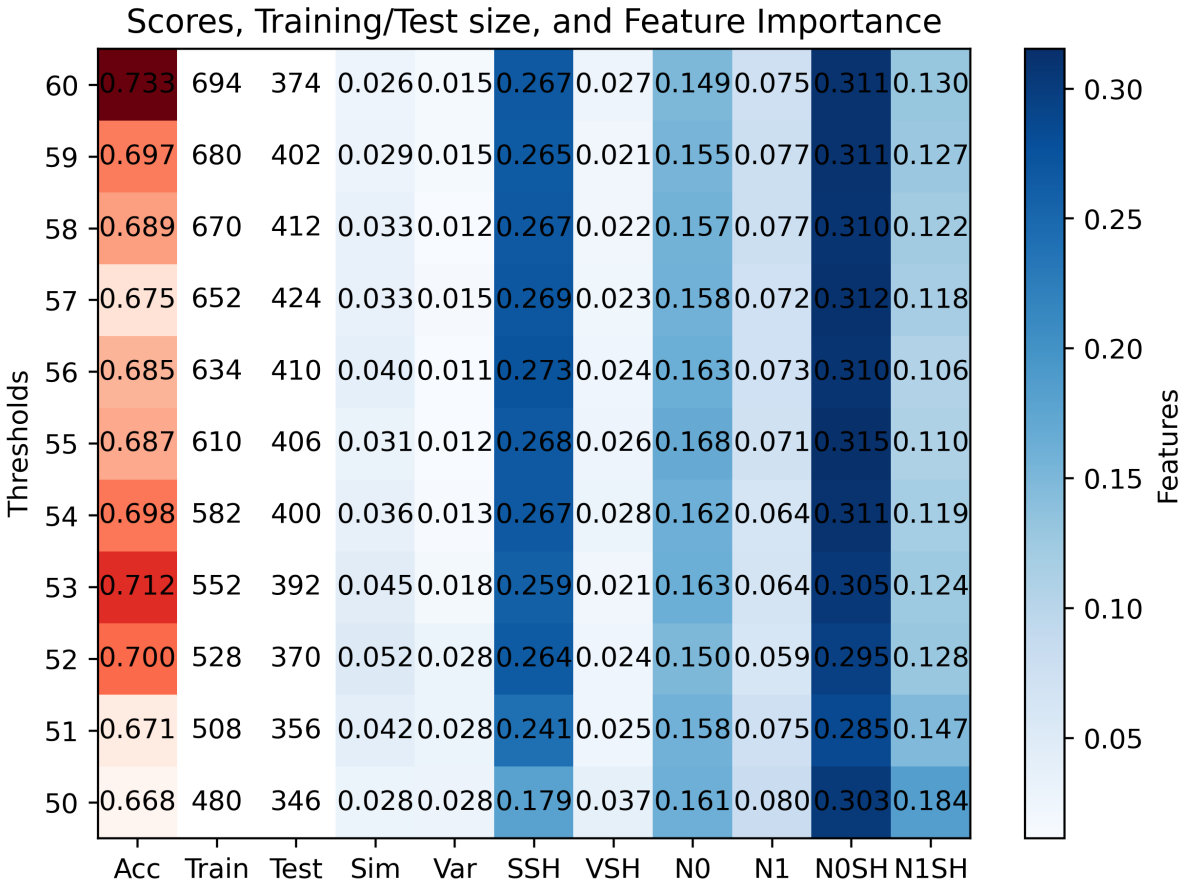

Figure 7: Classifying well-prediction on TestSetB. The classifiers were trained using sequences from TrainSetB and then tested using sequences from TestSetB.

the addition of auxiliary data have different effects on sequences of TrainSetA and TrainSetB. The effect of adding auxiliary information on the profile graphs of TrainSetB seems to be on the number of vertices (most likely reducing the number of competing alternates) while for the profile graphs of TrainSetA seems to be on the 1-dimensional holes (likely resolving conflicts). Finally, we note that the similarity after SHAPE (SSH) feature was important for both types of classifiers. This suggests that this feature captures the effect of the auxiliary data on the competing substructures, making it useful for this type of classification task.

4.2 Classifying directability

We built classifiers that take an RNA sequence as input and predict whether the sequence is directable (DIR) or not directable (ND). For these classifiers, we considered an RNA sequence as directable if the improvement in accuracy after adding auxiliary information is higher than a directability threshold. Otherwise, the sequence is classified as not directable. We varied the directability thresholds from 25% to 50%, these are given in the y-axis of Figures 8 and 9.

From Figure 8, which corresponds to classifiers trained in sequences in TrainSetA and tested in sequences in TestSetA, we see that for the task of directability, the most important feature is the similarity after SHAPE (SSH). And, from Figure 9, which corresponds to classifiers trained in sequences in TrainSetB and tested in sequence in TestSetB, we see that for directability, the most important features are the similarity

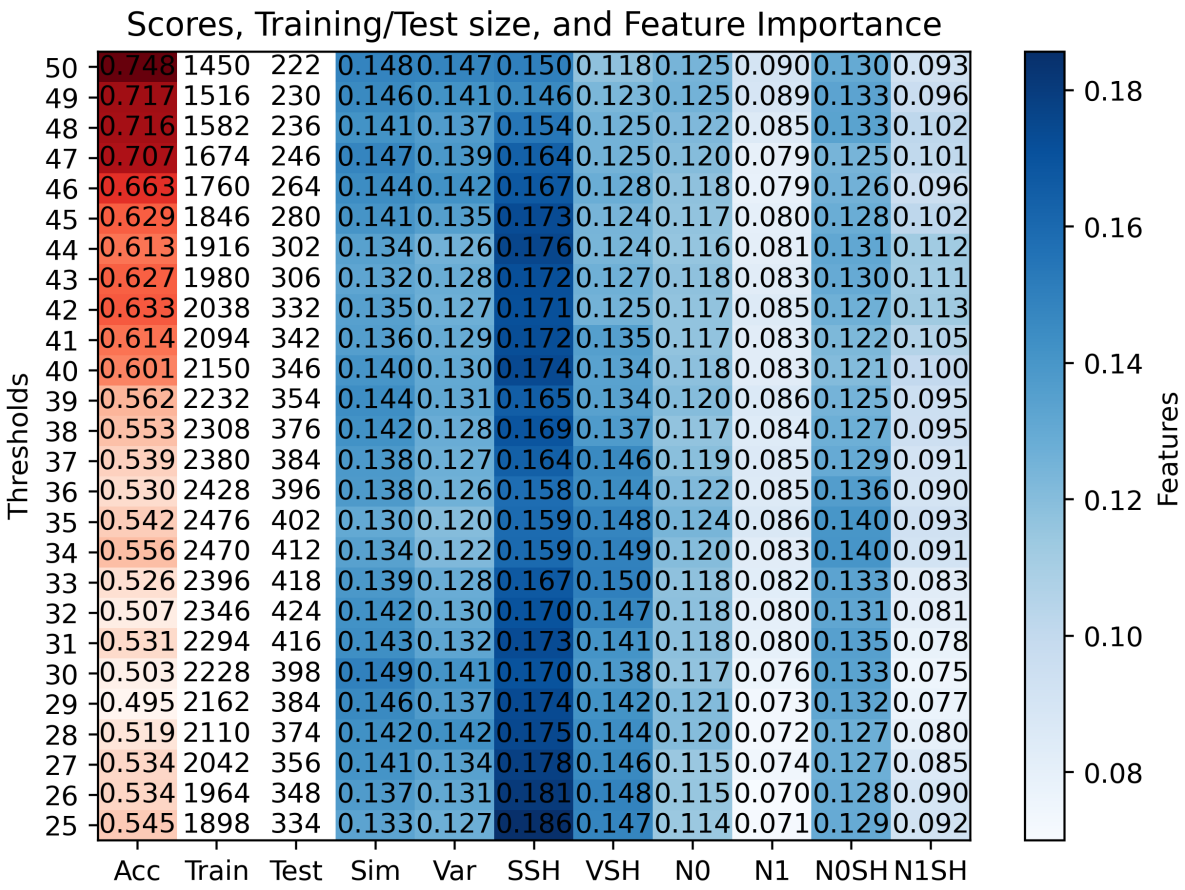

Figure 8: Classifying directability on TestSetA. The classifiers were trained using sequences from TrainSetA and then tested using sequences from TestSetA.

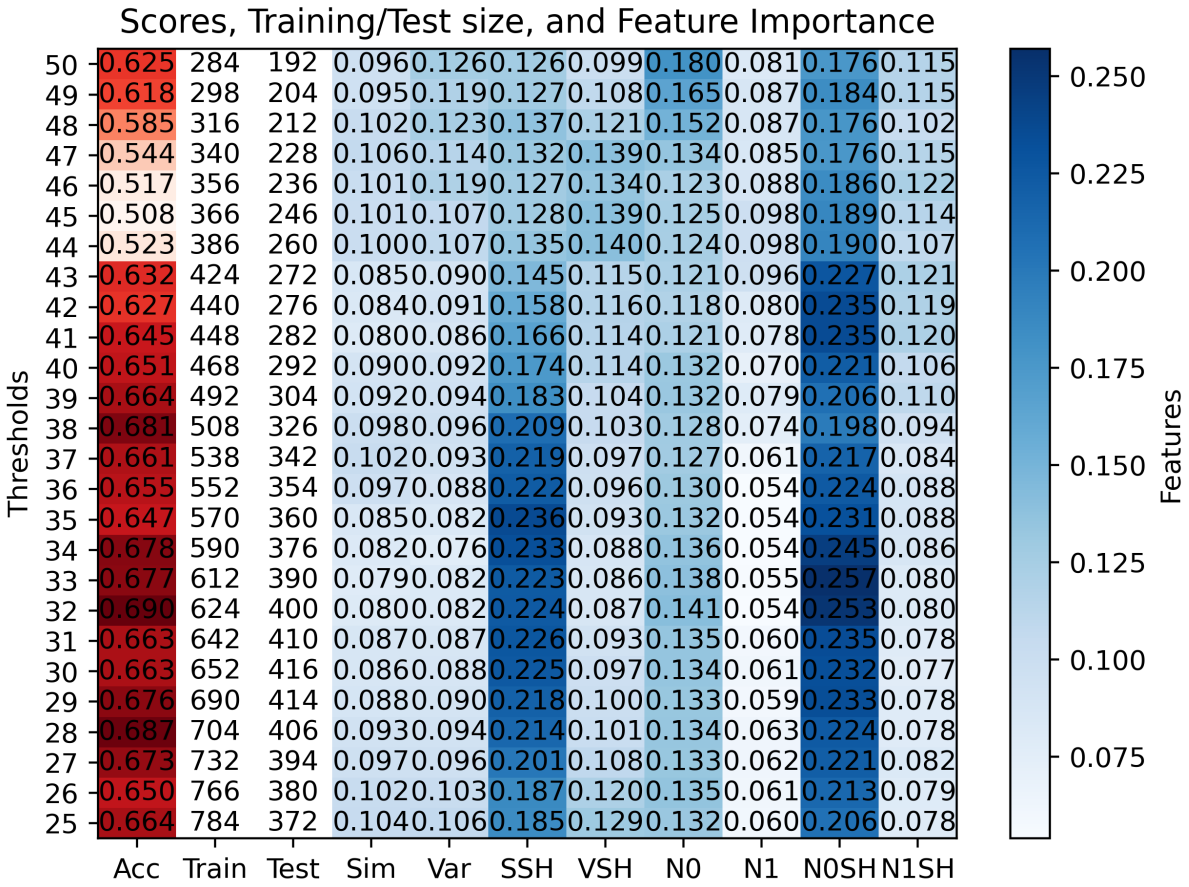

Figure 9: Classifying directability on TestSetB. The classifiers were trained using sequences from TrainSetB and then tested using sequences from TestSetB.

after SHAPE (SSH) and the 0-norm after SHAPE (N0SH). Similar to the case with the well-prediction classifiers, the difference in feature importance between classifiers trained in TrainSetA and classifiers trained in TrainSetB is likely due to differences in the composition of these two training sets of sequences. We note that the sequences in TrainSetB seem harder to direct (compared to the sequences in TrainSetA) as shown in Figures 4-5. Additionally, we observe that the N0SH feature captures the changes in size of the profile graphs, therefore it correlates with the number of competing profiles. Therefore, the fact that the N0SH feature is more important for classifiers trained in TrainSetB than for classifiers trained in TrainSetA (for both well-prediction and directability) highlights the differences in the effects of auxiliary information for directability when considering training sets with dissimilar sequences. Finally, as was the case for well-prediction, the similarity after SHAPE feature (SSH) was important for directability classifiers, indicating that this feature captures the effect of the auxiliary data on the competing substructures, making it useful for this type of classification task.

4.3 Predicting both accuracy and directability

Here we consider a case where we first predict well-prediction and then the directability of a sequence. Building this pipeline of classifiers was motivated by the idea that training on subclasses of sequences with similar properties (e.g., being already well-predicted or directable) improved the accuracy of the classifiers.

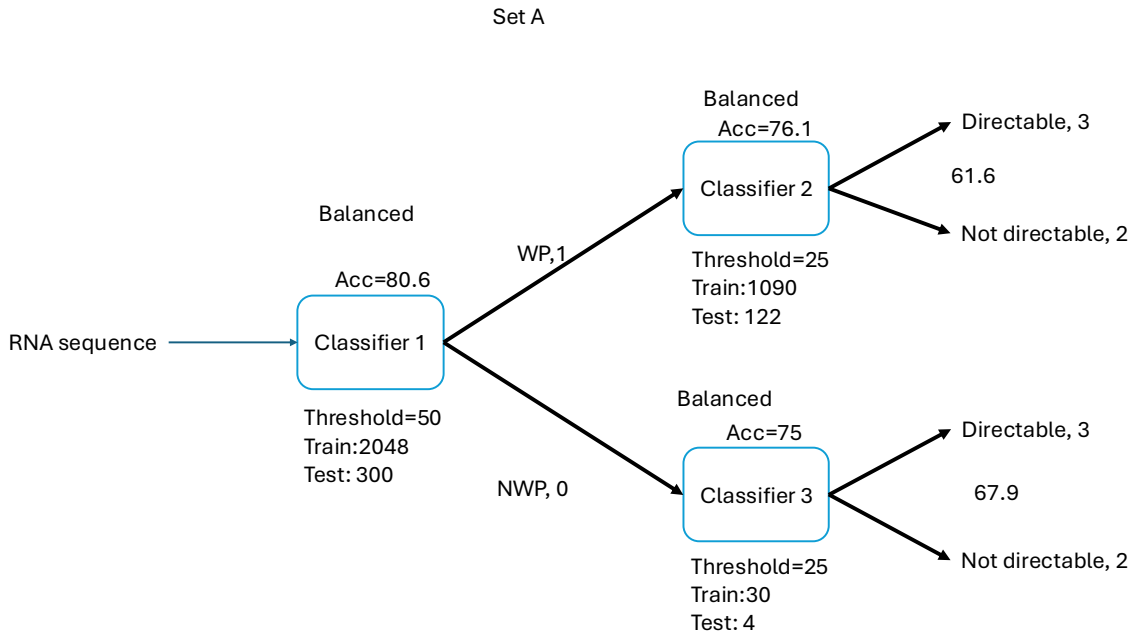

Figure 10: Classifying well-prediction and then directability on TestSetA. The classifiers were trained using sequences from TrainSetA and then tested using sequences from TestSetA. The labels for Classifier 1 are well-predicted (WP, 1) and not-well-predicted (NWP, 0) while the labels for Classifiers 2 and 3 are directable (DIR, 3) and not-directable (ND, 2). The training sets for each classifier have been balanced with equal number of sequences for each label.

Thus, we trained classifiers on sequences that are already well-predicted and on sequences that are not well-predicted. The pipelines in Figures 10-11 consider three classifiers. The first classifier is for well-prediction. This is similar to the classifier that we considered in Section 4.1. The other two classifiers (classifiers 2 and 3) are similar to those considered in Section 4.2, but these classifiers were trained only on sequences that are already well-predicted (classifier 2) and on sequences that are not well-predicted (classifier 3).

For example, in Figure 10, we first split the training set into WP and NWP sequences. We then classify the WP sequences according to directability to obtain Classifier 2. Similarly, we classify the NWP sequences according to directability to obtain Classifier 3. This establishes a two-step pipeline to classify sequences first according to well-prediction and then directability. To test this pipeline, we consider a balanced test set of n=300 sequences (150 WP, 150 NWP). Note that this is the test set in which Classifier 1 yields the greatest accuracy. First, we use Classifier 1 to classify each sequence in the test set as WP or NWP. Next, we only consider those sequences which were predicted as WP by Classifier 1 and classify these according to directability with Classifier 2. This yields an accuracy of 61.6 percent. Similarly, we consider those sequences which were predicted as NWP by Classifier 1 and classify these according to directability using Classifier 3. This gives us an accuracy of 67.9 percent.

The classifiers in Figure 10 were trained in TrainSetA and tested on sequences in TestSetA. Similarly, the classifiers in Figure 11 were trained in TrainSetB and tested on sequences in TestSetB where we balanced the training and test sets to have an equal number of sequences for each label. In both cases, we balanced the training and test sets so that there is an equal number of sequences for each label.

Here we note that restricting the training sets to sequences with similar properties (e.g., only well-predicted sequences) allows us to achieve higher accuracies as shown in the accuracies of the middle sub-classifiers in Figures 10-11. Although, we achieved higher accuracies on the subclassifiers of the pipelines of

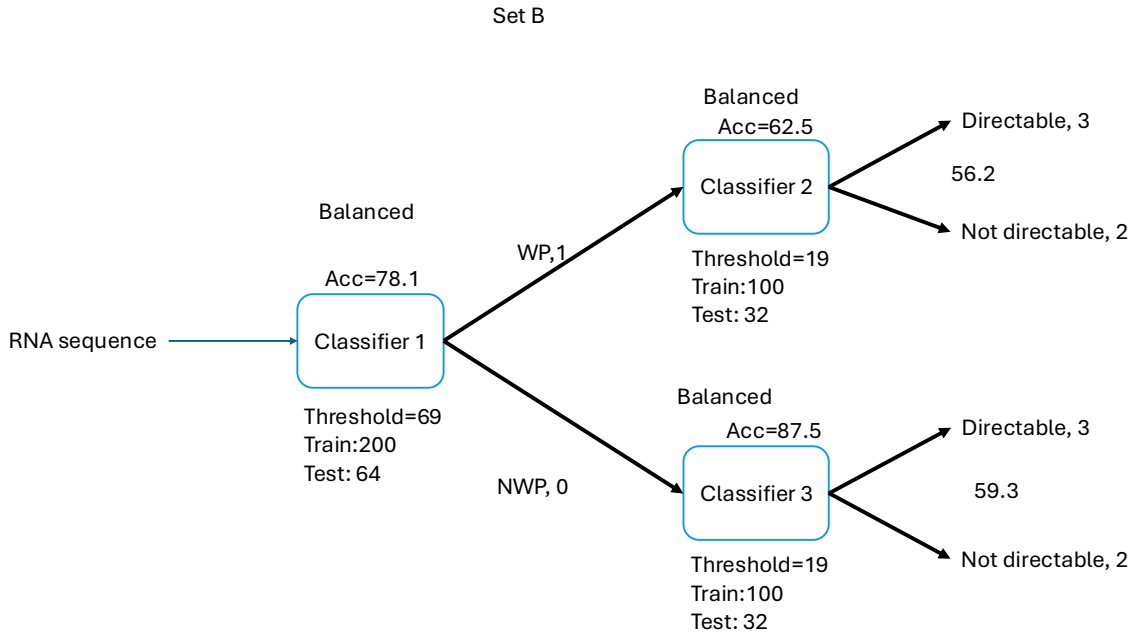

Figure 11: Classifying well-prediction and then directability on TestSetB. The classifiers were trained using sequences from TrainSetB and then tested using sequences from TestSetB. The labels for Classifier 1 are well-predicted (WP, 1) and not-well-predicted (NWP, 0) while the labels for Classifiers 2 and 3 are directable (DIR, 3) and not-directable (ND, 2). The training sets for each classifier have been balanced with equal number of sequences for each label.

Figures 10-11, the overall accuracy of each pipeline was lower than the direct classifier for directability.

5 Conclusions

In this paper, we considered features based on suboptimal formations competing with the minimum-free energy formation and investigated their role in determining well-prediction and directability. First, we introduced a similarity measure among competing substructures called profiles which was used to generate features for Random forest classifiers. Second, we presented an n-dimensional representation of the profiles which allows the use of topological data analysis (specifically, persistence landscapes) to obtain different metrics that represent topological features. Third, we built random forest classifiers using these similarities and topological features. Fourth, we show that the similarity features are more important for classifiers trained on sequences with similar structures, while the topological features are more important for classifiers trained on sequences with dissimilar structures. Finally, we performed extensive testing on two sets of RNA sequences where we studied the sensitivity of the classification accuracy and their feature importance. Overall, we identified the similarity after SHAPE (SSH) and the 0-norm after SHAPE (N0SH) are the most important features for directability and well-prediction. We also showed that one can achieve higher accuracies by restricting the training sets to sequences with similar properties such as training only on sequences that are well-predicted.

6 Acknowledgments

Part of this work was supported by a pilot grant (to D.M., L.S.G, and L.R.) from the University of Kentucky AI/ML Hub initiative. D.M. was partially supported by a Collaboration grant (# 850896) from the Simons Foundation and a grant (NSF: # 2424633) from the National Science Foundation. D.N. was partially supported by the National Science Foundation (NSF: # 2516126, # 2151802, and # 2245903).

7 Code and Data Availability

Python code in Jupyter notebooks that we used for training Random Forest classifiers are available on a GitHub repository at `https://github.com/LoganRose25/RNA-project`. We also provide the required data for training and testing these classifiers in csv files in the same repository. For persistence landscape computations done in R, the TDA-tools package is required. This R package and installation instructions can be found here: `https://github.com/jjbouza/tda-tools`. All other data supporting the findings of this study are available within the cited papers and their supplementary information.

References

[1] K. Darty, A. Denise, and Y. Ponty, "Varna: Interactive drawing and editing of the rna secondary structure," *Bioinformatics*, vol. 25, no. 15, p. 1974, 2009.

[2] E. Rogers and C. E. Heitsch, "Profiling small RNA reveals multimodal substructural signals in a boltzmann ensemble," *Nucleic Acids Research*, vol. 42, pp. e171–e171, nov 2014.

[3] K. Sato, M. Akiyama, and Y. Sakakibara, "Rna secondary structure prediction using deep learning with thermodynamic integration," *Nature communications*, vol. 12, no. 1, pp. 1–9, 2021.

[4] D. H. Turner and D. H. Mathews, "Nndb: the nearest neighbor parameter database for predicting stability of nucleic acid secondary structure," *Nucleic acids research*, vol. 38, no. suppl_1, pp. D280–D282, 2010.

[5] J. S. Reuter and D. H. Mathews, "Rnastructure: software for rna secondary structure prediction and analysis," *BMC Bioinformatics*, vol. 11, p. 129, 2010.

[6] R. Lorenz, S. H. Bernhart, C. Höner zu Siederdissen, H. Tafer, C. Flamm, P. F. Stadler, and I. L. Hofacker, "Viennarna package 2.0," *Algorithms for molecular biology*, vol. 6, pp. 1–14, 2011.

[7] N. R. Markham and M. Zuker, "Unafold: software for nucleic acid folding and hybridization," *Bioinformatics: structure, function and applications*, pp. 3–31, 2008.

[8] M. S. Swenson, J. Anderson, A. Ash, P. Gaurav, Z. Sükösd, D. A. Bader, S. C. Harvey, and C. E. Heitsch, "Gtfold: enabling parallel rna secondary structure prediction on multi-core desktops," *BMC research notes*, vol. 5, pp. 1–6, 2012.

[9] D. Layton and R. Bundschuh, "A statistical analysis of rna folding algorithms through thermodynamic parameter perturbation," *Nucleic acids research*, vol. 33, no. 2, pp. 519–524, 2005.

[10] E. Rogers, D. Murrugarra, and C. Heitsch, "Conditioning and robustness of rna boltzmann sampling under thermodynamic parameter perturbations," *Biophysical journal*, vol. 113, no. 2, pp. 321–329, 2017.

[11] K. A. Wilkinson, R. J. Gorelick, S. M. Vasa, N. Guex, A. Rein, D. H. Mathews, M. C. Giddings, and K. M. Weeks, "High-throughput shape analysis reveals structures in hiv-1 genomic rna strongly conserved across distinct biological states," *PLoS biology*, vol. 6, no. 4, p. e96, 2008.

[12] K. E. Deigan, T. W. Li, D. H. Mathews, and K. M. Weeks, "Accurate shape-directed rna structure determination," *Proceedings of the National Academy of Sciences*, vol. 106, no. 1, pp. 97–102, 2009.

[13] S. Washietl, I. L. Hofacker, P. F. Stadler, and M. Kellis, "Rna folding with soft constraints: reconciliation of probing data and thermodynamic secondary structure prediction," *Nucleic acids research*, vol. 40, no. 10, pp. 4261–4272, 2012.

[14] Z. Sükösd, M. S. Swenson, J. Kjems, and C. E. Heitsch, "Evaluating the accuracy of shape-directed rna secondary structure predictions," *Nucleic acids research*, vol. 41, no. 5, pp. 2807–2816, 2013.

[15] D. Willmott, D. Murrugarra, and Q. Ye, "Improving rna secondary structure prediction via state inference with deep recurrent neural networks," *Computational and Mathematical Biophysics*, vol. 8, no. 1, pp. 36–50, 2020.

[16] M. Szikszai, M. Wise, A. Datta, M. Ward, and D. H. Mathews, "Deep learning models for rna secondary structure prediction (probably) do not generalize across families," *Bioinformatics*, vol. 38, no. 16, pp. 3892–3899, 2022.

[17] Q. Zhao, Z. Zhao, X. Fan, Z. Yuan, Q. Mao, and Y. Yao, "Review of machine learning methods for rna secondary structure prediction," *PLoS computational biology*, vol. 17, no. 8, p. e1009291, 2021.

[18] M. Justyna, M. Antczak, and M. Szachniuk, "Machine learning for rna 2d structure prediction benchmarked on experimental data," *Briefings in Bioinformatics*, vol. 24, no. 3, p. bbad153, 2023.

[19] Y. Ding and C. E. Lawrence, "A statistical sampling algorithm for rna secondary structure prediction," *Nucleic acids research*, vol. 31, no. 24, pp. 7280–7301, 2003.

[20] N. Otter, M. A. Porter, U. Tillmann, P. Grindrod, and H. A. Harrington, "A roadmap for the computation of persistent homology," *EPJ Data Science*, vol. 6, no. 17, 2017.

[21] H. Edelsbrunner, D. Letscher, and A. Zomorodian, "Topological persistence and simplification," *Discrete & Computational Geometry*, vol. 28, no. 4, pp. 511–533, 2002.

[22] P. Bubenik *et al.*, "Statistical topological data analysis using persistence landscapes.," *J. Mach. Learn. Res.*, vol. 16, no. 1, pp. 77–102, 2015.

[23] P. Bubenik, "The persistence landscape and some of its properties," in *Topological Data Analysis: The Abel Symposium 2018*, pp. 97–117, Springer, 2020.

[24] E. Rivas, R. Lang, and S. R. Eddy, "A range of complex probabilistic models for rna secondary structure prediction that includes the nearest-neighbor model and more," *RNA*, vol. 18, no. 2, pp. 193–212, 2012.