2016

RNA-protein interaction prediction: String-based versus feature-based models

Maen Allaga

Follow this and additional works at: <https://researchrepository.wvu.edu/etd>

Recommended Citation

Allaga, Maen, "RNA-protein interaction prediction: String-based versus feature-based models" (2016). *Graduate Theses, Dissertations, and Problem Reports*. 5073.
<https://researchrepository.wvu.edu/etd/5073>

This Thesis is protected by copyright and/or related rights. It has been brought to you by the The Research Repository @ WVU with permission from the rights-holder(s). You are free to use this Thesis in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you must obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/ or on the work itself. This Thesis has been accepted for inclusion in WVU Graduate Theses, Dissertations, and Problem Reports collection by an authorized administrator of The Research Repository @ WVU. For more information, please contact researchrepository@mail.wvu.edu.

RNA-PROTEIN INTERACTION PREDICTION: STRING-BASED VERSUS FEATURE-BASED MODELS

by

Maen Allaga

**Thesis submitted to the
Benjamin M. Statler College
of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of**

**Master of Science
in
Computer Science**

Approved by

**Dr. Donald Adjero, Committee Chairperson
Dr. Elaine Eschen
Dr. Gianfranco Doretto**

**Lane Department of
Computer Science and
Electrical Engineering**

**Morgantown, West Virginia
2016**

**Keywords: Pattern Matching, n-grams, String algorithms,
Suffix trees, sequence, secondary structure, RNA, protein
Copyright 2016 Maen Allaga**

ABSTRACT

RNA-Protein Interaction Prediction: String-based versus Feature-based Models

Maen Allaga

Cellular processes are significantly influenced by the interactions between different RNAs and proteins within cells. This interaction is crucial in understanding gene expressions and gene regulations, and their role in various diseases. Empirical and experimental methods to study this interaction are hampered by the high cost and combinatorial nature of the problem. Consequently, computer science and machine learning methods were applied to predict the interaction between RNAs and proteins.

RNAs are sequences of nucleotides, while proteins are sequences of amino acids. The protein secondary structure describes how amino acids are positioned in three dimensional space. Early methods predicted the interaction between RNA and protein using only sequence information. Recent methods have shown the significance of secondary structure in understanding RNA-Protein interactions.

In this thesis, we explore prediction models for RNA-Protein interaction using two different schemes. The first applied string algorithms to extract the most effective string patterns from both sequences and secondary structures. This method resulted in a 93.39% prediction accuracy. The second method used a feature-based approach by combining extracted features from both sequences and secondary structures. The feature-based approach enhanced the prediction accuracy as it included much more available information resulting in a 94.77% accuracy.

Table of Contents

ABSTRACT	ii
Introduction	2
Background & Related Work	5
Feature-Based Approach	14
3.1 Introduction	14
3.2 Representing RNAs	14
3.3 Representing Proteins	17
3.4 Feature space and classification	19
1. Support Vector Machine (SVM)	20
2. Random Forest	21
String-Based Approach	22
3.1 Introduction	22
3.2 Suffix trees for protein and RNA strings	22
3.3 Richness for protein and RNA substrings.....	25
Results	32
3.4 Introduction	32
3.5 Datasets.....	32
3.6 Tools Used	33
3.7 Performance Measurement	34
3.8 Setting Parameters for Classification Algorithms.....	34
3.9 Experimental Results	38
3.10 Comparative results	42
Conclusion & Future Work	44
3.11 Conclusion	44
3.12 Future Work	44
References	46

Introduction

Cellular processes are determined to a good extent by the interactions between different Ribonucleic Acids (RNA) and Proteins within cells. This interaction is crucial in understanding gene expressions and gene regulations. RNA interaction with malfunctioning proteins has been implicated in cell misregulation, leading to serious diseases [1].

Empirical and experimental methods used to study this interaction can not cover all the wide range of different RNAs and Protein. A cell contains many RNAs and protein. With the possibility for a protein binding with an RNA, practical wet-lab biological experiments to study all the possible combinations of interacting proteins and RNAs becomes almost impossible, given the time and resources required. Computer science methods can be applied to help in this problem, based on a small set of known interacting RNA-Protein complexes.

From a biological point of view, a protein consists of a set of amino acids that are positioned together in a specific form in a three dimensional space. This creates complex structures that vary to characterize the protein. On the other hand, an RNA is a sequence of nucleotides that combine to create RNA structures defined by the energy needed for these nucleotides to form pairs.

Although Protein-RNA interaction prediction is known to be an important cellular process, we still lack a complete understanding of the characteristics of a protein and an RNA that allow them to interact, forming new complexes. Various laboratories are studying the interactions for specific pairs of protein and RNA, even without the ability to define the general properties that allowed such interaction to happen. Starting with experimentally known interacting pairs, computer scientists have attempted to help by applying machine learning methods to simplify the problem, trying to predict this interaction based on the already known information about both RNA and protein. [2]

The first step in this process is to define the best set of features that can best describe the RNA and protein with the available information, i.e. the amino acids and nucleotides. Sequences are used to represent both RNA and protein indicating the order of the amino acids and nucleotides for protein and RNA, respectively. Proteins include 20 different amino acids while RNA consists of 4 different nucleotides, which means that we can represent both protein and

RNA with sequences of an alphabet of the length 20 and 4 respectively [3]. Although sequence information describes both protein and RNA, experiments showed that secondary structures have an important impact on the interaction process [4], [5]. Therefore, some studies have started to include more information about secondary structures, combining information extracted from both sequences and secondary structures to obtain more relevant information and hence improve the prediction models.

The secondary structure describes how the molecules are bound together in a three dimensional space, and therefore, plays a crucial role in characterizing the interaction process. Secondary structures are mainly restricted by the energy required for the molecules to form a specific structure. Most secondary structure information is obtained by minimizing the energy required to form the structures. Computational methods have proved their effectiveness in extracting the secondary structure information.

RNA secondary structures describe how nucleotides are paired to form double stranded RNA in some area. Some methods have been developed to predict RNA secondary structures based on the nucleotide sequences. Thermodynamic models are the most popular amongst these methods [6]. They mainly rely on measuring free energy and building secondary structures on the assumption that minimum free energy formations are expected to be more stable [7].

On the other hand, protein secondary structures are defined by hydrogen bonds which determine how the amino acids are positioned in a three dimensional space. The protein secondary structure can be described in different ways. A popular approach is by using the dihedral angles (ϕ and ψ) between the amino acids. Dihedral angles define the angles of rotation between two planes, in this case, the planes are defined by the bonds between three adjacent amino acids [8]. Ramachandran codes are derived from Ramachandran plots which are plots describing the dihedral angles. They aim to reduce the Ramachandran plot into clusters based on distributions of the dihedral angles [9]. Protein blocks are another method used to describe protein secondary structures by considering the fold formed by five consecutive amino acids and then clustering these folds [10].

Earlier methods for RNA-Protein interaction prediction focused on sequence data, building features based on only sequence, for RNA and protein individually, or combining both

sequences and extract representative features [3]. Later, secondary structures were shown to be important in the process and therefore are now being included in the prediction process [4].

In this work, we consider different representations for both protein and RNAs. We considered sequence and structural information for both protein and RNA. For the sequences, we use the traditional 4-letter sequences for RNA, but a reduced 7-letter alphabet for protein. For structures, we use the Ramachandran codes for protein structure representation. These have been used in previous work on studying protein structures [8], where Ramachandran codes proved their information richness. To our knowledge, this work represents the first time Ramachandran codes are being used for protein-RNA interaction studies. RNA secondary structures have been used earlier in RNA alignment and Protein-RNA interaction prediction. In this work, we introduce the length of secondary structures element as a useful descriptor for RNA structures.

After defining the features, we built two different prediction models. The first approach extracted features from sequences and structures for protein and added to them features extracted from sequences and structures for RNA. The idea behind this model is to combine all the information about each protein and RNA individually, and then trying to use this information in the prediction problem.

The second approach is a string-based approach. Hence, we built a feature space based on q-grams (i.e., q-length substrings), also called q-mers in the biology literature. We analyzed the sequences of Protein-RNA pairs to determine the q-mers that tend to appear in interacting pairs, and those that are often found in non-interacting pairs. We then used these q-mer pairs as the feature set. We applied this approach on both sequence information and structure information after representing the structural information as strings.

The remainder of the thesis is organized as follows: Chapter 2 will provide some background and discuss previous work on predicting Protein-RNA interaction. In chapter 3, we present our first model which is feature-based. In chapter 4, we introduce a new string-based approach to RNA-protein interaction. We discuss both methods and compare their results in Chapter 5. Chapter 6 concludes the thesis.

Background & Related Work

The importance of RNA-Protein interaction comes from the key role it plays in regulating cellular processes. Researches have shown interest in RNA-Protein interactions primarily driven by the need to understand how cells work, including cell localization and other fundamental processes [4]. Furthermore, some studies have shown that some RNA-Protein interactions are related to some important diseases [1]. Ever since, work has been done to computationally predict the interaction between specific RNAs and Proteins. Clearly the problem of prediction is closely related to the problem of representing the RNA and proteins. Success in identifying and extracting the information that is most relevant to protein-RNA interaction will no doubt lead to improved prediction of such interaction.

In order to understand the problem of RNA-Protein interaction prediction, it is necessary to understand the nature of both RNA and protein and how they can be represented in a meaningful way. This step is necessary before the computer can understand and apply a prediction model based on this representation.

Proteins are a collection of amino acids bound together. There are twenty amino acids, forming a twenty-letter alphabet, namely Alanine (A), Cysteine (C), Aspartic acid (D), Glutamic acid (E), Phenylalanine (F), Glycine (G), Histidine (H), Isoleucine (I), Lysine (K), Leucine (L), Methionine (M), Asparagine (N), Pyrrolysine (O), Proline (P), Glutamine (Q), Arginine (R), Serine (S), Threonine (T), Selenocysteine (U), Valine (V), Tryptophan (W), Tyrosine (Y). In the simplest form, a protein can be described by the list of amino acids forming it. This is called a protein sequence or primary structure. It is the first representation that can be obtained for a protein, though, it is a powerful representation as it represents the core information about a protein representing the order of amino acids within the protein.

Proteins may contain any of the twenty amino acids, but due to the chemical similarity between amino acids, researchers usually group the amino acids set into different groups based on different criteria. One criteria used to group amino acids is based on their dipole moments and the volume of their side chains resulting in seven different groups [4]. Other grouping methods use their charge and polarity of side chains in RNA-Protein complexes. This grouping method will result four groups, namely {'DE'}, {'HRK'}, {'CGNQSTY'}, and {'AFILMPVW'} [11].

Although protein sequences contain core information about proteins, another meaningful information can be obtained about proteins. The protein secondary structure defines how the amino acids are positioned in a three dimensional space to form the protein shape. The importance of protein secondary structure comes from the crucial role of the three dimensional shape of a protein in molecular processes. It characterizes how a protein can interact with other molecules including other proteins, RNAs and DNAs [4].

Protein secondary structures take various forms. They describe how consecutive atoms are positioned in a 3D space. Amino acid residues are held together by hydrogen bonds (see Figure 2.1). One way to describe the protein structure is by using dihedral angles. A dihedral angle is formally defined as an angle between two planes, but in biology and chemistry, the dihedral angle is the angle between three consecutive atoms, in our case, three amino acid residues. These three angles are omega ω , phi ϕ and psi ψ . Omega angle is limited between 0 and 180 degrees, therefore, in most applications it is eliminated and only phi and psi are used. This notation is further expanded to different representations such as protein blocks or Ramachandran codes.

Figure 0.1 Protein Hydrogen bond [12]

Figure 0.2 The ϕ , ψ angular degrees of freedom in one residue of the protein backbone. The ω dihedral angle can be assumed to be fixed at 180° (trans) or 0° (cis) [13]

Figure 2.2 (taken from [13]) shows the three dihedral angles between amino acids. It shows the structure of an amino acid. An amino acid contains amine (NH_2) and carboxylic acid (COOH). The hydrogens used to bond amino acids together. Dihedral angles, as shown in Figure 2.2, exist between carbons, nitrogen and oxygen, denoted as C, N and O respectively. $\text{C}\alpha$ is the carbon atom next to the carboxyl group. While $\text{C}\beta$ is the next carbon atom in the amino acid [12].

The protein block is one way to represent protein 3D structures. It relies on coding the protein fold based on consecutive amino acids, i.e., 5-residue chains. De Brevern et al. developed a clustering algorithm to define different protein blocks. Their clustering algorithm identified 16 different protein blocks using five consecutive $\text{C}\alpha$ atoms [10]. That means protein blocks can be represented using a 16-character alphabet. In this representation the ω angles are neglected, only phi ϕ and psi ψ angles were considered, with the assumption that 5 consecutive residues will be enough to describe the protein secondary structures. After obtaining the dihedral angles for a dataset, a clustering method is applied to derive the 16 clusters representing the different protein blocks [10]. Figure 2.3 (taken from [10]) shows some samples representative folds using the protein blocks.

Figure 0.3 Description of protein blocks PBp, PBb, PBd and PBm [10]

In 1966, Ramachandran et al. [14] introduced what is now known as Ramachandran plots, which is a graph showing the distribution of dihedral angles. More specifically, they observed, that the allowed phi and psi angles that can occur in the hydrogen bonds restricts the combination of the angles that can be observed in practice. This means that for a specific dataset, there would be some regions in the phi/psi plot that will be empty while other regions will have a higher density (see Figure 2.4) [14].

Figure 0.4 Ramachandran plot

The Ramachandran codes are derived from the Ramachandran plots by clustering the inhabited areas in the plots and assigning each area some symbol. Then we can derive a new

representation for protein secondary structures by encoding each phi and psi angles with the cluster code where they belong based on the Ramachandran codes. The number of clusters is subject to the dataset under consideration, and the purpose of the encoding. It often varies between different researchers based on the application and the clustering method [8]. Ramachandran codes provides a simple, efficient, and yet effective way to represent protein secondary structures. It uses symbol sequences similar to protein sequences to capture secondary structures information represented by the dihedral angles.

On the other hand, RNA (ribonucleic acid) is a set of nucleotides which have essential biological roles including coding, decoding and expressing genes. RNA contains four different nucleotides, namely guanine, uracil, adenine and cytosine denoted G, U, A and C respectively. Similar to protein sequence, RNA sequence provides the very basic representation for RNAs. It provides a listing of the nucleotides that form the RNA in order. Despite its simplicity, RNA sequence is the root to all other RNA representations.

Though RNA is single stranded, it can fold such that some part of it will create a double stranded RNA. The basic rule for this folding is tying a U nucleotide to A, and G nucleotide to C and vice versa. This folding forms the secondary structure of an RNA. Given the wide variety of RNAs, it is too difficult to examine all RNAs and find their secondary structure. Thus, different prediction methods have been developed to predict RNA secondary structures, mainly by minimizing the free energy. When RNAs fold, nucleotides form pairs, thus to obtain a valid secondary structure, prediction methods often attempt to preserve the stable thermodynamic properties following minimal energy models [15].

Figure 0.5 RNA Folding

RNA secondary structure can be expressed using a simple representation called the dot-bracket notation, it is a balanced parenthesis notation where a single nucleotide is represented by a dot, and double stranded nucleotides are represented by open and closing brackets depending on their position, the nucleotide that appears first in the sequence is the open bracket and the corresponding complementary nucleotide is the closing bracket [16]. Dot-bracket notation is the simplest notation for RNA secondary structure and more meaningful notation can be derived based on it. When RNA folds and nucleotides form pairs, we can differentiate different shapes, often called the secondary structure elements (SSE). These SSEs when combined with protein secondary structures could play a significant role in determining the possibility of interaction between specific RNA and protein. Prior studies defined diverse types of SSEs including Stem, Hairpin loop, Internal loop and Bulges. Other more complicated elements can be derived from these basic elements, however, studies showed that the simple elements can still provide important information to aid RNA alignments and interaction prediction [16]. Figure 2.6 shows example SSEs and their combination to derive more complicated structures.

Figure 0.6 RNA secondary structure elements

The Protein-Protein interaction problem is well studied due to the importance of proteins in all cell processes, including gene coding and decoding. Shen et al. [17] were among the first to predict protein-protein interaction using only sequence information. They used a simplified 7-character alphabet to represent protein sequences and built a Support Vector Machine (SVM) prediction model using a high quality database consisting of 16,443 entries experimentally validated entries. They were not only able to predict Protein-Protein interaction, but also to build a protein interaction network that shows the relationship and connections between different proteins based on their interactions [17]. For other related protein-protein interaction see [18], [19], [20].

However, the problem of Protein-RNA interaction prediction has also been addressed in some previous work. Earlier attempts used only sequence information, as secondary structure information were not readily available. Recent results [4] have shown that different types of information about RNA and protein secondary structure could lead to a significantly improved prediction performance.

Muppirala et al. [3] explored RNA and protein sequences separately. They build a feature space of 599-dimensions with 343 features extracted from Protein and 256 features extracted from RNA. The 343 protein features were extracted by first considering 7 groups used in by Shen et al. [17] representing 20 amino-acid based on dipole moments and the volume of their

side chains. To conserve locality information, the notion of triads was used to extend the feature space to $7 \times 7 \times 7$ features for protein and $4 \times 4 \times 4$ features for RNA. Two classification models were deployed to build the prediction scheme, namely, an SVM and a Random Forest model. The models were trained using two different datasets RPI369 and RPI2241. See Section 5.2 for more detailed description on datasets used in this and other work. The Random Forest model trained with RPI2241 obtained the most accurate results among other trained models in this work, achieving 89.6% in accuracy with 0.89 and 0.90 for Precision and Recall respectively.

In [11] Wang et al. applied q-mers approach by finding the pairs of protein amino acids and RNA nucleotides that tends to appear together. They worked with a reduced protein alphabet, the 20 amino acids were grouped based on their charge and polarity. The new alphabet consisted of 4 groups representing the 20 amino acids. Then, they considered Protein 4-mers and RNA 3-mers, this consideration allowed them to preserve some of the locality information which is claimed to have high impact on the prediction process. The feature space consisted of 4094-dimensions space (4^3 features for Protein, and 4^3 feature for RNA). This high dimensional space requires relatively large datasets for training, Thus, they extracted 500 features that have the highest impact on the prediction. The Naive-Bayes- classifier was used and they tried different datasets including RPI369, RPI2241 and NPInter. The accuracy achieved for the three datasets were 75%, 74% and 77.6%, respectively.

RPI-Pred [4] was developed by Suresh et al. to predict the interaction between non-coding RNA and proteins. They included information from both sequences and secondary structures, building a feature-vector of 132 features. Hence, 20 features describe the RNA considering 4 different amino acids and 5 secondary structure elements including stem, hairpin, loop, bulges. They used 112 features to characterize proteins, following the 7-letter alphabet for amino acid sequences, and the extracted 16 protein blocks [10]. They extracted a new dataset namely RPI1807 and used this for training. They tested their method using RPI369, RPI2241 and NPInter. SVM scheme was used to build the prediction model. They were able to achieve an accuracy of 92%, 84% and 86.9% using RPI369, RPI2241 and NPInter, respectively.

Lu et al. [5] followed a different approach in predicting RNA-Protein interaction. The core difference in their work is in the way features are extracted. RNA secondary structures were predicted using Vienna RNA package. Additionally, they used Hydrogen bonding information,

then using the Fourier transformation they extracted a feature set for both RNA and protein to form the feature vector for each RNA and protein. They included the first ten terms of the Fourier series for each information type. This work was limited to long non-coding RNAs, therefore, their prediction model cannot accurately predict the interaction for all the different kinds of RNAs. They built a training dataset contains 649 non-redundant protein-RNA pairs. 322 of the pairs are interactive pairs, and the rest 327 are non-interactive pairs. They trained a scoring matrix which gives a score for each protein-RNA pair. Based on the assigned score, they predict the interaction between protein and RNA. They tested the method using NPInter dataset and they were able to achieve 77% in accuracy.

Recently, the problem of RNA-Protein interaction has been considered from the viewpoint of complete structural representations. Zhang et al. [21] developed a deep learning model to define the preferences of RNA Binding Protein (RBP) structural representations. They used information from predicted RNA tertiary structures to address the problem of RPI. This helped them to define a three dimensional representation for RPI complexes, which they claimed to have its own structures that can decide the binding preferences.

For both RNA and protein, their sequences and secondary structures can each be expressed in the form of strings [5], [22], [23], [24]. In this work, we take advantage of efficient string algorithms and data structures to extract suitable string-based feature sets and build a prediction model that can predict the interaction between RNAs and proteins. The other contribution is to build a prediction model based on protein Ramachandran codes and improved RNA secondary structure representations. The prediction models used in this work are Support Vector Machines and Random Forests to explore the influence of classification schemes on the prediction performance.

Feature-Based Approach

3.1 Introduction

In this work we will use information for both sequences and secondary structures. This applies to both the feature-based approach discussed in this chapter and string-based approach we will introduce next.

Features are the attributes that will be used in prediction. We will build our feature space based on the count of different elements captured from sequences and secondary structures. In this chapter, we will introduce our RNA and protein representations. Then we will define our feature space as the count or distribution of certain characteristics from the presented representations.

3.2 Representing RNAs

For RNA sequences, although a nucleotide can be any of A, U, C and G nucleotides, some RNAs are not completely known and they include X at some positions denoting that the nucleotide at this position is unknown, therefore the alphabet for RNA sequences is extended to 5 characters (A, U, C, G, X). This alphabet will be used to encode RNA sequences in this work.

The RNA secondary structure is more complicated. The information about RNA secondary structure is not always available for all RNAs, thus, many methods have been developed to predict the RNA secondary structure based on RNA sequences. Most of these methods use energy-based models that predict secondary structure by folding RNA sequences with the goal to minimize the free energy and preserve thermodynamic properties such as stability. RNA folding methods often represent RNA secondary structures using the dot-bracket notation. This is the simplest secondary structure notation where double stranded nucleotides are represented by open and closing brackets and other nucleotides are represented with dots, resulting in strings with balanced parentheses.

One of the most famous RNA folding programs is Vienna RNA Secondary Structure server [7]. It is developed as a set of services implementing RNA secondary structure prediction (Folding), RNA Alignment, and RNA Inverse folding, i.e., finding an RNA sequence that folds into a specific secondary structure [7]. Vienna RNA package 2 includes many more services,

such as, finding distance matrices and calculating the thermodynamic distance between two secondary structures and more. Vienna RNA is widely used for RNA folding, thus in this work, we used it to predict RNA secondary structures. RNAFold is the service within Vienna RNA that predicts RNA secondary structures. The method implements a free energy model to predict the secondary structure for a given RNA sequence.

As noted earlier, the dot-bracket notation is the simplest notation for secondary structures and more meaningful notations can be derived based on it. In our case we will use secondary structure elements (SSE). For simplicity, we consider four different elements, namely Stem, Loop, Internal Loop and Bulges (See Figure 3.1).

Figure 0.1 Secondary Structure Elements: 1) Stem of 5 nucleotides 2) Loop of 6 nucleotides 3) Internal Loop of 6 nucleotides 4) Bulge of 3 nucleotides

As shown in Figure 3.1, a stem is any number of consecutive pairs of nucleotides. This is mainly what creates the secondary structures. In the dot-bracket notation, stems form the open and closing brackets. Unpaired nucleotides form all other kinds of secondary structure elements. Unpaired nucleotides that appear at the edge of a stem form a loop. These correspond to the dots surrounded by opening and closing bracket in a dot-bracket notation. Bulges on the other hand are unpaired nucleotides that appear within one end of a stem. They are called left bulges or right bulges based on their location. An internal loop can be seen as a special case of bulges, where they appear on both sides of a stem at the same position. This classification can help in taking RNA secondary structures one step further and includes better information regarding RNA

interaction with protein, as the secondary structure helps to better define interaction probabilities [4].

Moreover, studies suggested that RNAs with different lengths can have different functional roles [16], [25]. Hence, we studied the length distribution for secondary structure. Based on these, we classify each SSE into three different categories: long, medium and short. Figure 3.2 shows the probability distribution for the lengths of the SSEs using 1,078 RNA sequences from the RPI1807 dataset. This provides more important additional information about secondary structure elements than just considering only the secondary structure elements directly. The cutoff for each length class was related to the distribution in the first place, mainly trying to equally distribute the population over the three classes, and taking into consideration the overlaps between lengths (see Table 3-1).

Figure 0.2 Distribution of the length of RNA Secondary Structure Elements

Table 0-1 RNA Secondary structure element classes

	Stem	Bulge	Loop	Internal Loop	Not classified
Short range	1-3	1-1	1-4	1-2	1-2

Medium range	4-5	2-2	5-6	3-5	3-5
Long range	6-	3-	7-	6-	6-

3.3 Representing Proteins

Proteins consist of amino acids which can be any of the 20 amino acids. Given the chemical similarity between amino acids, we can group amino acids into different groups and use the groups instead of the individual amino acids to represent protein sequences. Amino acids grouping can be done based on different criteria. In this work, we follow Shen et al. [17] who suggested classifying amino acids by their dipole moments and the volume of their side chains, resulting in seven groups, as indicated in Table 3-2.

Table 0-2 Amino acid groups

Group	Amino Acids
1	Alanine A, Glycine G, Valine V
2	Isoleucine I, Leucine L, Phenylalanine F, Proline P
3	Tyrosine Y, Methionine M, Threonine T, Serine S
4	Histidine H, Asparagine N, Glutamine Q, Tryptophan W
5	Arginine R, Lysine K
6	Aspartic acid D, Glutamic acid E
7	Cysteine C

The protein secondary structure is defined by the positions of consecutive molecules in 3D space. The positions can be described by the dihedral angles between each three consecutive molecules, denoted by omega ω , phi ϕ , and psi ψ , respectively (see Figure 3.3).

Figure 0.3 Dihedral angles ϕ , ψ and ω between consecutive amino acids within a protein [13]

As discussed earlier, due to the limitation in ω angle, usually only phi ϕ and psi ψ angles are considered in describing the protein secondary structure. Ramachandran et al. [14] studied the relation between phi and psi angles and presented Ramachandran plots, showing the density of the joint occurrences of phi and psi (see Figure 3.4). This density can be used to derive protein secondary structure representation by clustering the values from the Ramachandran plot then replacing the values of phi, psi pairs by their cluster representation. In this work, we will use a 7-character alphabet representing 7 clusters from the Ramachandran plot. Hence, the protein representation used will consist of a 7-character alphabet representing amino acid groups (for the sequence), and another 7-character alphabet for secondary structure represented by Ramachandran codes.

Figure 0.4 Ramachandran Plot

3.4 Feature space and classification

Now that we have the sequences and secondary structures for RNA and proteins well defined, we can construct our feature space. Each nucleotide either belongs to any of the four secondary structure elements or doesn't belong to any, which makes it five choices for secondary structure elements. Each secondary structure element might have three different length classes, that makes it five nucleotides $\times$ five secondary structures elements $\times$ three different classes, or a total of 75 features representing an RNA. Protein sequences are reduced to seven groups resulting a 7-character alphabet. Secondary structures are also represented in seven different Ramachandran code clusters for each position. Thus, the protein representation requires seven characters for sequence $\times$ seven different Ramachandran codes, making it 49 features. The complete feature vector for an RNA-Protein pair will thus be 124 features counting each of the different variations of sequence information and secondary structure information.

The power of this representation comes from the fact that it includes all the available information using a small number of features. That is crucial in building the prediction model in a reasonable time and space, and to ensure reasonable accuracy in the prediction.

Building the feature vector and finding the best representation for prediction model is just the first step in RNA-Protein interaction prediction. The next step is to decide what prediction scheme is to be used. Machine Learning methods are divided to two main groups, supervised learning and unsupervised learning. The difference lies in the prior knowledge. In supervised learning, we provide the algorithms with a test dataset and the algorithms find the relation between the input and output within the dataset. In unsupervised learning, we provide the input and the algorithm decides the output labels. It can also be used to identify hidden patterns within the data by finding related records or pairs of data.

The RNA-Protein interaction prediction problem is to determine whether or not a pair of protein and RNA will interact. Therefore, we already know our classes, thus, we are dealing with a supervised learning, specifically, a classification problem. Many different algorithms have been developed to solve classification problems. For example, Nearest Neighbors (NN) algorithms decide on the class of a data point by looking at its neighbors and consider the point has the same class as the majority of its neighbors. Decision Trees are hierarchical models that classify data points based on their features. During the classification process, the leaves mean we reached a class, while internal nodes mean the model is not yet determined regarding the class and more features should be considered at the lower levels [26].

More classification methods have been developed. However, considering the Protein-RNA interaction problem, SVM and Random Forest proved to be the best methods for building classifiers with highest accuracy. Therefore, we will consider these two methods and compare the performance of both classifiers.

1. Support Vector Machine (SVM)

Support Vector Machine (SVM) algorithm is a statistical model that tries to infer a hyper plane that best separates data points belonging to two different classes. That is, the plane splits the feature space and distinguishes the classes [27, 28].

Different SVM formulations have been developed. For instance, C -SVM finds the hyper plane with the largest margin between training patterns and decision boundary, and can be used in classification problems with two classes [29]. ν -SVM, on the other hand, includes optimizing one more parameter, namely ν , proved to be an upper bound on the fraction of margin errors [30].

In general, the problem of SVM classification can be expressed by finding the function $f(x)$:

$$f(x) = \text{sign} \left(\sum_{i=1}^l y_i \alpha_i K(x_i, x) - b \right)$$

where x is the input vector representing data points, α is the Kuhn-Tucker coefficient, b is an offset threshold. K is a transformation function, which transforms the input vector into the feature space [31]. The value of α changes the weight given to different vectors, hence, its bound is the first parameter to be defined when finding $f(x)$. The formula of the function K includes another important parameter, namely γ [31].

In this work, we will explore the performance of SVM under different values of C and γ , in order to obtain the best classification result.

2. Random Forest

Random Forest is a classification method where independent decision tree classifiers are built based on different set of features, and all these trees vote for their classes and the class that gets the most votes would be considered for the data point at hand [32].

The performance of random forest algorithm relies on the performance of the decision tree classifier included.

As for decision tree classifier, it is affected by the depth of the tree, the number of trees, and the number of features included in each tree. In this work, we will not limit the depth of the trees as the tools used can handle the feature space. The only drawback here would be the time consumed to build the model, which is not a problem in our case as we will build the model offline.

String-Based Approach

3.1 Introduction

Although the feature-based approach is powerful and contains a lot of information about RNA and protein, it does disregard important local information. This local information could be important in the prediction process. The fact is that when an RNA bonds to a protein, it is not just an amino acid and a nucleotide that are involved, but a set of neighboring nucleotides against a set of neighboring amino acids. Furthermore, the secondary structures should be compatible to allow RNA-Protein bonding. These observations suggest that we could consider small portions or q-grams of sequences and structures when building the feature vector rather than looking for individual molecules.

This will typically lead to a very large dimensional feature space. Consider for instance, the 5-mer strings under a 75-character alphabet (using the RNA representation that we presented in Chapter 3). This means we have more than $75^5 \approx 2 \times 10^9$ different 5-mers to consider for RNA only, besides the $49^5 \approx 2 \times 10^8$ for proteins. This is a huge feature space that is very difficult to handle with the current computation limitations. Consequently, we need to reduce our feature space to a reasonable dimensionality. A quick observation is that this will be very sparse, as most combination of the RNA and protein symbols will not occur in practice.

Reducing feature space dimensionality means we need to select some of the k-mers and drop the rest. Not all k-mers strings hold the same amount of information, thus, to enhance prediction results, we need to identify the k-mers that have the most significant influence on the prediction of interacting or non-interacting pairs. Hence, we look for the k-mer strings that appear most in positive pairs and k-mers strings that appear most in negative pairs, and then construct a feature vector based on these.

3.2 Suffix trees for protein and RNA strings

The naive approach to find the most occurring substrings is to count all of them within the dataset. The running time for this algorithm is $O(n*k)$ for each k-mer or $O(n^2*k)$ for all k-mers, where n is the total number of characters in the dataset. This means we need to hold a large dictionary of all possible substrings, the size of this dictionary would be of order $O(|\Sigma|^k)$ where

$|\Sigma|$ is the alphabet size (in our case $|\Sigma| = 75$ for RNA and $|\Sigma| = 49$ for protein). We need to find a better approach to study the distribution of k-mers, i.e. a memory and time efficient method to find the k-mers that contribute most in the interaction process.

An improved approach will be to go over the database to first determine all the k-mers that actually occurred, and then use standard linear-time pattern matching algorithms to determine their respective number of occurrences. This will still require an overall $O(k \times n^2)$ time, worst case.

Suffix trees and suffix arrays [23], [24], [33], [34] provide a better tool to find the most occurring substrings within a sequence. Let S be a string of length n , $S = \{s_0, s_1, s_2, \dots, s_n\}$. The suffix of S that starts at position k is $S_k = \{s_k, s_{k+1}, s_{k+2}, \dots, s_n\}$. A suffix tree is a data structure that consists of all suffixes of a string [35]. Ukkonen developed an online construction algorithm that builds a suffix tree using suffix links in linear time and linear space. The idea is to update all suffix links while scanning the string S which takes only one scan for each symbol of S to build the whole suffix tree in $O(n)$ time [24]. Figure 4.1 shows the suffix tree for an example sequence.

The suffix tree has been used in searching and matching genomic data due to its efficiency in both space and time. It provides a simple structure to sort long genome sequences using a simple tree navigation process. Furthermore, matching two genomes can be done using suffix trees by building the suffix tree for a genome and then trying to match the second genome with the built tree [36].

Figure 0.1 The suffix tree for $S = \text{acracca\$}$ [24]

Using suffix trees, we can find the count and distribution of all substrings of any length by adding count value to each node while building the suffix tree. To find the count for each

substring, we concatenate all RNA strings together in one string then we use the suffix tree of this whole string to find the most occurring substrings and use them to build the feature vector.

For example, in figure 4.1, let's consider a 3-mer "acr". We start navigating the tree at the root. Then we will go to the "a" branch. Then, we go to branch "c". And finally we go down to branch "r" to obtain the count of this 3-mer. The look-up takes $O(k)$ time. Notice that the count at each node is calculated while constructing the suffix tree, i.e. no additional time is required to calculate the count over the time required to build the suffix tree, which is $O(n)$.

The suffix tree requires an $O(n)$ time and space for construction. After construction, we can traverse the $O(n)$ nodes of the suffix tree to determine the occurrence counts of each substring in $O(n)$ time. Thus, this is the time required to count the k-mers in the string, for a given k.

The power in using the suffix tree to find the distribution of k-mers is that we don't need to maintain a large dictionary. We can even do it in constant space if we want to consider a fixed length substring. For instance, we investigate substrings up to the length of five characters so we need a tree with the depth of five levels at most. The suffix tree can be used to manage all substrings of all lengths without extra storage, as they are covered by nodes in lower levels. For example, if we want to count substrings of length four, we need to navigate the tree up to level four only. The other benefit of using suffix trees is to enhance the space required to manage all substrings count. In the dictionary model, for every entry in the dictionary, we need to store 5 characters for that specific entry. With the suffix tree, we store only one character at each level reducing the total number of characters by 80%. So we did not only reduce the number of entries in the dictionary, but we also reduced the number of characters stored. On the other hand, when counting substrings, we can ignore some branches of the tree based on the count of their parent node. For example, if we reach a node representing a substring, say "aguu", and we found out the count to be less than our threshold to consider this substring, then we don't need to navigate to its children as they will have a count that is at most the same as their parent. Thus, we can disregard the whole branch. This observation makes a big difference when it comes to filtering substrings based on their count, especially, given that we have a large tree to navigate.

We have built suffix trees counting occurrences of each substrings of length 2 to 5 characters for RNA sequence and secondary structure, protein sequence and secondary structure.

That is, we constructed four suffix trees, one for each type of string representation we used, namely: RNA sequence, RNA secondary structure elements represented as strings, protein sequence, protein secondary structure represented as a sequence of Ramachandran codes.

3.3 Richness for protein and RNA substrings

It makes more sense to count occurrences in both positive pairs and negative pairs so we can find how a specific k-mer contributes in the positive or negative pairs. K-mers that tend to occur more in positive pairs hold more information deciding a positive RNA-Protein pair than other k-mers that appear equally in both positive and negative pairs. Similarly, k-mers that appear more in negative pairs tend to contribute more in classifying an RNA-protein pair as negative pair. This observation implies that, we should look for more than just occurrence count. The richness could be a better attribute to express a k-mer contribution in the interaction between pairs.

Given a k-mer, let:

γ_+ = number of occurrence of the k-mer in positive pairs

γ_- = number of occurrence of the k-mer in negative pairs

The k-mer richness is simply defined as: $R = \frac{\gamma_+}{\gamma_-}$

Thus, k-mers with richness greater than 1 appear more in positive pairs (positive k-mers). While a richness value near zero means we have a negative k-mer. Richness values close to 1 are associated with k-mers that appears equally in both positive and negative pairs, hence they hold no information in the sense of interaction prediction.

Given the foregoing, we now construct four suffix trees for the positive dataset, and another four suffix trees for the negative dataset. We extracted pairs that appeared only in the positive pairs, or only in the negative pairs as they hold the most information. Their distribution showed that we have enough feature to rely on those that appeared in one of the two datasets only.

Using the dataset RPI1807, we built suffix trees of depth five and we found distributions for k-mers up to length five. RNA 4-mers appeared for as many as 18,020 times in positive pairs and 16,894 in negative pairs, while the number drops for 5-mers to 3,700 appearances in positive pairs and 3,494 in negative pairs. Figures 4.2 to 4.5 show distributions of every k-mer of length

1 to 5 in RNA and protein positive and negative pairs. Obviously, the numbers drop as the k-mer length increases. Protein 4-mers occurred about 1,134 times in positive and 443 times negative pairs. Figures 4.2-4.5 only show k-mers distribution over the truncated positive and negative datasets, given the large number of possible k-mers.

Figure 0.2 RNA positive k-mers count (for k=1 to 5)

Figure 0.3 RNA negative k-mers count (for k=1 to 5)

Figure 0.4 Protein positive k-mers count (for k=1 to 5)

Figure 0.5 Protein negative k-mers count (for k=1 to 5)

As mentioned, the direct k-mers count distribution may not be the best way to capture the information carried by the k-mers. Thus, we considered the richness as the main factor to compare k-mers and build the feature vector. See Figures 4.6 to 4.11. RNA k-mer richness varied in the range of 0.36 to 3, This gives a very tight bound to filter the k-mers, therefore, we set a positive richness threshold of 1.5 and negative threshold of 0.6. Protein richness, on the other hand, had more variation. It ranged between ∞ (where k-mers appeared mainly in positive pairs) to 0.01 (where k-mers appeared mainly in negative pairs). This distribution makes it easier to choose k-mers by selecting the ones that appears in positive/negative pairs based on their counts.

We set the threshold for positive k-mers to 67, and to 0.03 for negative k-mers. The corresponding log plots shows the distribution more clearly.

Figure 0.6 RNA k-mers richness

Figure 0.7 RNA k-mers richness filtered greater than 1.5 (positive k-mers) or less than 0.6 (negative k-mers)

Figure 0.8 RNA k-mers richness (log values)

Figure 0.9 Protein k-mers richness

k-mers with protein sequence k-mers, and RNA structure k-mers with protein structure k-mers. For sequence data, the feature space is of size $7^5 \times 5^5 \approx 10^7$, and for secondary structures the space would be $7^5 \times (5 \times 3)^5 \approx 10^{10}$.

The same analysis applied before is applied to this data, and the results were promising as we were able to distinguish pure positive k-mers and pure negative k-mers in both sequence data and secondary structure data. We further applied count threshold to decrease the size of the feature space. We set sequence thresholds to 200 for positive k-mer pairs and 150 for negative k-mers. This resulted in 4,680 k-mers pairs consisting our feature vector. For secondary structures we extracted 2,350 k-mers by setting positive k-mer threshold to 55 and negative k-mer threshold to 20.

Once we obtained the described feature vectors, we applied both SVM and Random Forest classification methods on three different feature vectors. The first one included 4,680 features from sequence only, the second feature vector contained 2,350 features representing only secondary structures, while the last experiment included 7,030 features combining both sequence and structure.

This method showed promising results which will be discussed in the next chapter. Its power comes from relying on including sequence and secondary structure data, combining both RNA and protein data together, building feature vectors based on these, and finally by preserving information on locality through the of consecutive molecules as one unit.

Results

3.4 Introduction

In this thesis we presented two different RNA-Protein interaction prediction schemes. The first scheme used features extracted from sequences and secondary structures, while the second scheme applied suffix trees to extract k-mers of length up to 5 and deploy them as feature vector in order to preserve locality information.

The feature-based approach included data from every molecule in the RNA and protein. Data about sequence and secondary structure were combined to produce a combined feature space. Features representing RNA and protein are extracted separately, 75 features for RNA and 49 features for protein. The feature space is relatively small, though powerful. Including RNA secondary structure element size aimed to focus more on secondary structure which helps to further enhance the prediction accuracy as more information are included.

The string-based approach on the other hand used a much larger feature space of 7,030 features, with 4,680 of them extracted from RNA and protein sequences and the rest 2,350 extracted from RNA and protein secondary structures. The larger feature space captured local information, thus, we find regions of RNA and protein that tend to bond and allow the pair to interact. This concept is more precise than just looking for individual molecules as interactions involve sets of adjacent molecules. We extracted k-mers from different combinations, from RNA sequence and protein sequence (denote this set as QQ), RNA secondary structure and protein secondary structure (SS), RNA sequence and protein secondary structure (QS), and finally RNA secondary structure and protein sequence (SQ). The aim of using different combinations is to capture the most discriminating k-mers regardless of their source.

3.5 Datasets

Machine learning algorithms performance relies heavily on dataset quality, specifically, in classification problems. It is important to obtain valid dataset with both positive pairs and negative pairs. In biology, this quality depends on the availability of experimental data. The Protein Data Bank (PDB), Nucleic Acid Database (NDB) and Protein-RNA Interface Database (PRIDB) are widely known databases that provide Protein-RNA complexes which can be parsed

to create a dataset. Suresh et al. [4] parsed NDB and PDB to build their dataset. In their work they depended on atomic interaction between Protein and RNA pairs. That is, they used the atomic distance between RNA and protein sequences extracted from RNA-Protein complexes. Using a threshold on the atomic distance, they were able to separate strongly interacting pairs from weakly interacting pairs. Thus, RNA-Protein pairs with strong interaction are considered positive pairs, and pairs with weak interaction are considered negative pairs. The dataset they obtained was used to build classifiers which performed very well against different datasets. This dataset will be called RPI1807. It has 1,807 positive pairs and 1,436 negative pairs.

RPI369 and RPI2241 [3] are two RNA-Protein interaction datasets. Both datasets are non-redundant. As the name indicates, RPI369 contains 369 RNA-protein pairs. RPI2241 contains 2,241 RNA-protein pairs. Muppirala et al. extracted RPI369 and RPI2241 from RPIDB. RPIDB is a database that contains RNA-protein complexes extracted from PDB [37]. RPI2241 contains rRNA, ncRNA or mRNA. RPI369 was extracted from RPI2241. It contains only non-ribosomal complexes [3].

In this work, we will first use the RPI1807 dataset to measure the performance of our methods. To compare the results from our classifiers and prediction methods against previous work, we will use RPI369 and RPI2241 datasets, in addition to the RPI1807 dataset.

3.6 Tools Used

Many machine learning tools were developed and trusted by researchers. One of these tools is Weka [38]. Weka is an open source collection of machine learning algorithms implementations. It provides implementations for classification, clustering, regression and other algorithms. It also provides data pre-processing methods beside results visualization. Weka is written in Java, which allows calling Weka from our Java source code. It is possible also to run Weka as a stand-alone application with its simple GUI. In this work, we used Weka to perform classification using SVM and Random Forest. We used the current version of Weka which is Weka 3.6.13.

3.7 Performance Measurement

The performance of our approaches was evaluated using 10-fold cross-validation. In 10-fold cross-validation, we split the training dataset into 10 subsets of the same size. We trained the model 10 times covering every possibility. Each time we used the first 9 subsets as the training dataset and the 10th subset for testing. To measure the performance, we used accuracy, precision, recall and F-measure defined as follows:

$$Accuracy (ACC) = \frac{tp + tn}{tp + tn + fp + fn} * 100$$

$$Precision (PRE) = \frac{tp}{tp + fp}$$

$$Recall (REC) = \frac{tp}{tp + fn}$$

$$F - Measure (FSC) = \frac{2 * precision * recall}{precision + recall}$$

where: tp is true positive --the count of correctly classified positive pairs--, fp is false positive the count of wrongly classified positive pairs, tn is true negative denoting the count of correctly classified negative pairs, and fn is false negative the count of wrongly classified negative pairs. We calculated area under curve (AUC) which ranges between 0 and 1, with 1 indicating best possible performance.

3.8 Setting Parameters for Classification Algorithms

SVM algorithm has two main parameters, namely, C and γ , explained earlier. We ran SVM with the following values of C and γ

$$C = 2^i \text{ where } i \in \{-5, 15\}$$

$$\gamma = 2^i \text{ where } i \in \{-15, 3\}$$

We compared the values of accuracy and AUC for each values of C and γ . Figure 5.1 shows the performance of string-based SVM against different values of C . All this was done using linear SVM function. Figure 5.2 shows the performance against different values of γ . Using the string-based approach, SVM performed best with the cost value of 2^5 and γ value of 2^{-7} achieving of 93% and AUC of 0.93.

Figure 0.1 String-based SVM performance against cost values

Figure 0.2 String-based SVM performance against values of γ

For the feature-based approach, the best performance was reached when cost was set to 2^7 and gamma value set to 2^{-11} . Figures 5.3 and 5.4 shows feature-based approach's performance.

Figure 0.3 Feature-based SVM performance against cost values

Figure 0.4 Feature-based SVM performance against gamma values

Hence we used these values when evaluating the performance of our methods.

Random Forest has only one parameter to be evaluated, which is the number of random tree classifiers included. For this purpose, we tried different number of the classifiers. We tried the values of (10, 25, 50, 75, 100, 150, 200). Figures 5.5 and 5.6 represent the string-based approach and the feature-based approach respectively. The figures show that the String-based approach scored the best performance at with 200 random tree classifiers included.

Feature-based approach scored the best performance also using 200 random tree classifiers. Accuracy reached 94.77%, and AUC was at 0.983. Therefore, we will use 200 random tree classifiers in this work.

Figure 0.5 String-based Random Forest performance against number of random tree classifiers included

Figure 0.6 Feature-based Random Forest performance against number of random tree classifiers included

3.9 Experimental Results

We applied two different prediction methods, SVM and random forest. In some previous work, SVM performed better than random forest (e.g. RPI-Pred [4]), while in some others random forest outperformed SVM (e.g. Rise [3]). Thus, we applied both schemes to see their performance under our framework.

Table 5-1 compares the performance of our feature-based method using SVM and random forest prediction schemes. SVM method resulted in an accuracy of 94.61%. Precision, recall and F-measure were 0.95, 0.953 and 0.952, respectively. SVM showed a good performance. However, classification using random forest outperformed SVM, resulting in 94.77% in accuracy and 0.948 in precision, recall and F-measure.

Table 0-1 Performance of feature-based method

Measurements	Feature-Based	
	SVM	Random forest
AUC	0.945	0.983
PRE	0.95	0.948
REC	0.953	0.948
FSC	0.952	0.948
ACC	94.61	94.77

We applied the same tests using the string-based method. Table 5-2 shows the results using both SVM and random forest prediction methods. SVM prediction method scored an accuracy of 93%, with precision, recall and F-measure of 0.931, 0.93 and 0.93, respectively. The Random Forest method performed much better with accuracy of 93.3 9%, and 0.935, 0.934 and 0.934 in precision, recall and F-measure, respectively.

Table 0-2 Performance of string-based method

Measurements	String-Based	
	SVM	Random forest
AUC	0.926	0.984
PRE	0.931	0.935
REC	0.93	0.934
FSC	0.93	0.934

ACC	93	93.39
-----	----	-------

As with the feature-based method, the performance of random forests exceeded that of SVM, hence, we decided to use random forests as our main prediction method.

The string-based approach used different approach from the feature-based approach with respect to feature vector extraction. We tested k-mers extracted from different combinations of RNA sequence, RNA secondary structure, protein sequence and protein secondary structures.

Table 5-3 shows the top 30 k-mer pairs in positive dataset, and Table 5-4 shows top 30 k-mer pairs in negative dataset. Note that these k-mer pairs are encoded based on our representation using RNA and protein sequences and secondary structures (QSQS).

Table 0-3 Top 30 positive k-mer pairs

Protein k-mer	RNA k-mer	Richness (Log value)	Positive Count
DDCD	OOOO	∞	90
DDCD	BBBM	∞	86
DDCD	MBBB	∞	86
DDCD	MMBB	∞	86
DDCD	IAAA	∞	85
DCDF	BBBB	∞	77
DCDF	BBBD	∞	77
DCDF	BBDD	∞	77
DCDF	BDDD	∞	77
DCDF	DBBB	∞	77
DCDF	DDBB	∞	77
DCDF	DDDB	∞	77
DCDF	BBBE	∞	76
DCDF	BBBH	∞	76
DCDF	BBBJ	∞	76
DCDF	BBBL	∞	76
DCDF	BBEE	∞	76
DCDF	BBHH	∞	76
DCDF	BLLL	∞	76
DCDF	BEEE	∞	76
DCDF	BHHH	∞	76
DCDF	LLLL	∞	76
DCDF	CCCC	∞	76

DCDF	CCCI	∞	76
DCDF	CCII	∞	76
DCDF	CIII	∞	76
DCDF	DDDD	∞	76
DCDF	EBBB	∞	76
DCDF	EEBB	∞	76
DCDF	EEEB	∞	76

Table 0-4 Top 30 negative k-mers

Protein k-mer	RNA k-mer	Richness (Log value)	Positive Count
CADDD	CCMMB	$-\infty$	62
CADDD	CMMBB	$-\infty$	62
CADDD	HAAAF	$-\infty$	60
CADDD	CCGAH	$-\infty$	59
CADDD	CGAHH	$-\infty$	59
CADDD	BAACC	$-\infty$	57
CADDD	BBAAC	$-\infty$	57
CADDD	CJAAL	$-\infty$	57
CADDD	DAAAK	$-\infty$	57
CADDD	KAAAI	$-\infty$	57
CADDD	AAGAJ	$-\infty$	55
CADDD	AGAJB	$-\infty$	55
CADDD	BAGAA	$-\infty$	55
CADDD	BBAGA	$-\infty$	55
CADDD	CGAAE	$-\infty$	55
CDDDD	BGGAD	$-\infty$	55
CDDDD	DAGGB	$-\infty$	55
CDDD	CMMB	$-\infty$	53
CDDD	BAAC	$-\infty$	51
CDDDD	CCMMB	$-\infty$	51
CDDDD	CMMBB	$-\infty$	51
CDDD	BBMC	$-\infty$	50
CDDD	BMCC	$-\infty$	50
CDDDD	BAACC	$-\infty$	49
CDDDD	BBAAC	$-\infty$	49
CDDDD	CJAAL	$-\infty$	49
CDDDD	BBBMC	$-\infty$	48
CDDDD	BBMCC	$-\infty$	48
CDDDD	BMCCC	$-\infty$	48

CDDDD	KAAAI	$-\infty$	48
-------	-------	-----------	----

Table 5-5 compares result of prediction using k-mers extracted from RNA sequence and secondary structures and protein sequence and secondary structure (denoted as QSQS), k-mers extracted from RNA sequence and protein sequence (denoted as QQ), k-mer extracted from RNA secondary structure and protein secondary structure (denoted as SS), k-mers extracted from RNA sequence and protein secondary structures (denoted as QS) and k-mers extracted from RNA secondary structure and protein sequence (SQ).

Table 0-5 Performance of string-based method using different combinations sequence k-mers and structures k-mers

Measurements	String-Based method				
	QSQS	QQ	SS	QS	SQ
k-mers count	7,030	4,680	2,350	3,255	2,955
AUC	0.98	0.93	0.66	0.64	0.95
PRE	0.93	0.93	0.79	0.78	0.91
REC	0.93	0.93	0.67	0.65	0.89
FSC	0.93	0.98	0.61	0.58	0.89
ACC	93.35	93.19	66.98	65.15	89.29

As expected, using k-mers of RNA and protein sequences and secondary structures (QSQS), led to the best overall result. This is due to the comprehensive nature of the information using this combination. Interestingly, when using k-mers from only sequences (QQ), the method performed very close to that of using all available information. One reason could be that the secondary structure depends on the sequence information, and therefore the amount of information obtained from secondary structure might be limited. Using k-mers from only secondary structures (SS) dropped the performance to 66.98% in accuracy. Precision, recall and F-measures were 0.79, 0.65 and 0.58 respectively. The results showed that sequences hold the most decisive information. Secondary structures could add some information when combined with sequence information, but they seem to lack precision when used alone. However, adding the secondary structures seemed to significantly improve the AUC.

Using all RNA sequence and protein secondary structure (QS) scored the worst prediction performance, reaching accuracy of only 65.15%, though, precision reached 0.78,

meaning that this combination catches positive pairs rather than negative pairs. RNA secondary structures and protein sequences (SQ) performed well, reaching accuracy of 89.29%. Based on these results, we can conclude that the protein sequence provides the most important information needed to obtain good prediction performance with respect to RNA-protein interaction. From Table 5-5 above, we can infer that whenever protein sequence information is included, the prediction performance enhanced to reach at least 89% in accuracy, while the absence of this information leads to a significant drop in both the accuracy and AUC.

3.10 Comparative results

For comparative evaluation of our new methods, we compared our work with two recent approaches. RPI-Pred [4] used both sequences and secondary structures. It deployed RNA sequence and secondary structure elements, and used protein sequence and protein block as secondary structure representative. The other method is Muppirala et al. [3] which used only sequence information but they applied two different prediction methods: SVM and random forest. We will denote to Muppirala's SVM method as RPISeq-SVM and their Random Forest method as RPISeq-RF.

First, we compared the results using RPI2241 dataset –see Table 5-6. Our feature-based method had an accuracy of 88.97%, performing better than RPI-Pred (84%) and RPISeq-SVM (87.1). The accuracy was very close to RPISeq-RF which performed the best with accuracy of 89.6%. Other performance measures followed the same pattern with RPISeq-RF performing better than other methods with very small enhancement over our feature-based method. Precision, recall and F-measure scores were as described in Table 5-6. Interestingly enough, our feature-based method scored a better AUC (0.94) than RPISeq-RF (0.92), second to RPISeq-SVM (0.97).

String-based method performed less than feature based method. Accuracy, precision, recall and F-measure were all very close to RPISeq-SVM and outperformed RPI-Pred as described below.

Table 0-6 Comparative performance of RNA-protein interaction prediction methods using 10-fold CV on RPI2241

Measurements	RPI2241				
	Feature-Based	String-Based	RPI-Pred	RPISeq- SVM	RPISeq- RF

AUC	0.94	0.92	0.89	0.97	0.92
PRE	0.89	0.86	0.88	0.87	0.89
REC	0.89	0.86	0.78	0.88	0.90
FSC	0.88	0.86	0.83	0.87	0.90
ACC	88.97	86.52	84.0	87.1	89.6

On the other hand, when testing against RPI369 dataset, the performance of both RPISeq-RF and RPISeq-SVM dropped to an accuracy of 76.2% and 72.8% respectively. RPI-Pred on the other hand enhanced its performance, achieving an accuracy of 92.0%. Both our feature-based and string-based method outperformed all three methods achieving an accuracy of 97.88% and 96.38% respectively. All other measures for our methods scored over 0.95, better than each of the three previously proposed methods. Table 5-7 compares the results when using RPI369 dataset.

Table 0-7 Comparative performance of RNA-protein interaction prediction methods using 10-fold CV on RPI369

Measurements	RPI369				
	Feature-Based	String-Based	RPI-Pred	RPISeq- SVM	RPISeq- RF
AUC	0.99	0.98	0.95	0.81	0.81
PRE	0.97	0.96	0.89	0.73	0.75
REC	0.97	0.96	0.89	0.73	0.78
FSC	0.97	0.96	0.89	0.73	0.77
ACC	97.88	96.38	92.0	72.8	76.2

Compared to other studies, we showed better consistency when using different datasets. Our string-based approach scored accuracy of at least 86.52% over different datasets. The feature-based method scored at least 88.97% when testing against different datasets. We built our methods trying to cover as much RNA-Protein pairs as possible. Results showed that we succeeded in reaching our goal of creating a new RNA-protein interaction prediction scheme that can be reliable when used with different datasets, without frequently updating or changing the models, the selected features, or model parameters.

Conclusion & Future Work

3.11 Conclusion

RNA-Protein interaction (RPI) is an important topic when studying molecules and cells, as it affects many important cellular processes. The last few years, more RNA and protein complexes were discovered, and the need to find their interacting partners has grown. Given the huge cost and labor involved in experimental determination of RNA-protein interacting pairs, computational methods were developed to help with the problem. Most new methods are focused on specific RNA types, or they used limited information, leaving a large room for enhancement.

In this thesis, we proposed two different approaches to predict RPI. We included RNA secondary structure element size as an important component for describing RNA secondary structure elements, and used the Ramachandran codes to represent protein secondary structure. This is the first attempt to use this two pieces of information for the purpose of RPI prediction. Further, we developed an innovative k-mers approach deploying string-algorithms and suffix trees for the problem of RPI prediction. The string-based approach maintained local information which plays a crucial role in the interaction between RNA and protein. Our methods outperformed state-of-the art methods in the field of RPI prediction, showing more consistency over different datasets and different types of RNA and protein.

3.12 Future Work

Both developed methods in this thesis outperformed work done in the field, however, there is still space for improvement. Sequence representation can improve higher level information, protein tertiary information is already being verified, while methods to predict RNA tertiary information are being developed. Protein secondary structure can include protein blocks as additional information, although it is related to the Ramachandran codes. Yet the protein blocks could hold a different kind of information.

Feature-based approach might be enhanced if RNA information are combined with protein information. It could create a large feature space of 3,675 features, but the current computational resources handle this feature space, as we already managed to handle a feature space of 7,030 dimensions.

On the other hand, string-based approach can explore longer substrings and check the enhancement of using 6-mers or even longer substrings. Still, we doubt the effectiveness of this approach to enhancing the prediction. Perhaps, a better approach could be finding features from combined sequence and secondary structure data. We extracted k-mers from RNA sequences against Protein sequences, and RNA secondary structures against protein secondary structures. Combining RNA sequence with secondary structure information and protein sequence with secondary structure information and then extracting k-mers could enhance the performance, as this will consider all the information jointly, rather than considering each one on its own.

References

- [1] E. Jankowsky and M. E. Harris, "Specificity and nonspecificity in RNA–protein interactions," *Nature Reviews Molecular Cell Biology*, vol. 16, no. 9, pp. 533-544, 2015.
- [2] A. M. Khalil and J. L. Rinn, "RNA–protein interactions in human health and disease," *Seminars in Cell & Developmental Biology*, vol. 22, no. 4, pp. 359-365, 2011.
- [3] U. K. Muppirala, V. G. Honavar and D. Dobbs, "Predicting RNA-Protein interactions using only sequence information," *BMC Bioinformatics*, vol. 12, no. 1, p. 198, 2011.
- [4] V. Suresh, L. Liu, D. Adjero and X. Zhou, "RPI-Pred: predicting ncRNA-protein interaction using sequence and structural information," *Nucleic Acids Research*, vol. 43, no. 3, pp. 1370-1379, 2015.
- [5] Q. Lu, S. Ren, M. Lu, Y. Zhang, D. Zhu, X. Zhang and T. Li, "Computational prediction of associations between long non-coding RNAs and proteins," *BMC Genomics*, vol. 14, no. 1, p. 651, 2013.
- [6] M. Zuker and P. Stiegler, "Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information," *Nucleic Acids Research*, vol. 9, no. 1, pp. 133-148, 1980.
- [7] I. L. Hofacker, "Vienna RNA secondary structure server," *Nucleic Acids Research*, vol. 31, no. 13, pp. 3429-3431, 2003.
- [8] W.-C. Lo, P.-J. Huang, C.-H. Chang and P.-C. Lyu, "Protein structural similarity search by Ramachandran codes," *BMC Bioinformatics*, vol. 8, no. 1, p. 307, 2007.
- [9] R. W. Hooft, C. Sander and G. Vriend, "Objectively judging the quality of a protein structure from a Ramachandran plot," *Bioinformatics*, vol. 13, no. 4, pp. 425-430, 1997.
- [10] A. G. de Brevern, C. Etchebest and S. Hazout, "Bayesian probabilistic approach for predicting backbone structures in terms of protein blocks," *Proteins: Structure, Function, and Bioinformatics*, vol. 41, no. 3, pp. 271 - 287, 2000.
- [11] Y. Wang, X. Chen, Z.-P. Liu, Q. Huang, Y. Wang, D. Xu, X.-S. Zhang, R. Chen and L. Chen, "De novo prediction of RNA–protein interactions from sequence information," *Molecular BioSystems*, vol. 9, no. 1, pp. 133-142, 2013.
- [12] J. Clark, "Protein Structure.," October 2012. [Online]. Available: <http://www.chemguide.co.uk/organicprops/aminoacids/proteinstruct.html>. [Accessed 24 January 2016].
- [13] W. Boomsma, K. V. Mardia, C. C. Taylor, J. Ferkinghoff-Borg, A. Krogh and T. Hamelryck, "A generative, probabilistic model of local protein structure," *Proceedings of the National Academy of Sciences*, vol. 105, no. 26, p. 8932–8937, 2008.
- [14] G. N. Ramachandran, C. M. Venkatachalam and S. Krimm, "Stereochemical criteria for polypeptide and protein chain conformations: III. Helical and hydrogen-bonded polypeptide chains," *Biophysical Journal*, vol. 6, no. 6, p. 849–872, 1966.
- [15] I. L. Hofacker, M. Fekete and P. F. Stadler, "Secondary structure prediction for aligned RNA sequences," *Journal of Molecular Biology*, vol. 319, no. 5, pp. 1059-1066, 2002.
- [16] E. Mattei, G. Ausiello, F. Ferrè and M. Helmer-Citterich, "A novel approach to represent

- and compare RNA secondary structures," *Nucleic Acids Research*, vol. 42, no. 10, pp. 6146-6157, 2014.
- [17] J. Shen, J. Zhang, X. Luo, W. Zhu, K. Yu, K. Chen, Y. Li and H. Jiang, "Predicting protein–protein interactions based only on sequences information," *Proceedings of the National Academy of Sciences*, vol. 104, no. 11, pp. 4337-4341, 2006.
 - [18] G. C. K. W. Koh, P. Porras, B. Aranda, H. Hermjakob and S. E. Orchard, "Analyzing protein–protein interaction networks," *Journal of Proteome*, vol. 11, no. 4, p. 2014–2031, 2011.
 - [19] T. Nepusz, H. Yu and A. Paccanaro, "Detecting overlapping protein complexes in protein–protein interaction networks," *Nature Methods*, vol. 9, pp. 471-472, 2012.
 - [20] C. Reynolds, D. Damerell and S. Jones, "ProtorP: a protein–protein interaction analysis server," *Bioinformatics*, vol. 25, no. 3, pp. 413-414, 2008.
 - [21] S. Zhang, J. Zhou, H. Hu, H. Gong, L. Chen, C. Cheng and J. Zeng, "A deep learning framework for modeling structural features of RNA-binding protein targets," *Nucleic Acids Research*, vol. 44, no. 4, p. e32, 2016.
 - [22] R. Beal and D. Adjeroh, "Efficient pattern matching for RNA secondary structures," *Theoretical Computer Science*, vol. 592, pp. 59-71, 2015.
 - [23] D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*, Cambridge University Press, 1997.
 - [24] D. Adjeroh, T. Bell and A. Mukherjee, *The Burrows-Wheeler Transform: Data Compression, Suffix Arrays, and Pattern Matching*, Springer, 2008.
 - [25] D. Patterson, K. Yasuhara and W. Ruzzo, "Pre-mRNA secondary structure prediction aids splice site prediction," 2002.
 - [26] L. Rokach and O. Maimon, "Introduction to decision trees," in *Data Mining with Decision Trees*, vol. 69, Tel Aviv, World Scientific, 2007, pp. 1-11.
 - [27] Y. Ma and G. Guo, *Support Vector Machines Applications*, Springer, 2014.
 - [28] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273-297, 1995.
 - [29] B. E. Boser, I. M. Guyon and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, New York, NY, 1992.
 - [30] P.-H. Chen, C.-J. Lin and B. Schölkopf, "A tutorial on v-support vector machines: Research articles," *Applied Stochastic Models in Business and Industry - Statistical Learning*, vol. 21, no. 1, pp. 111 - 136, 2005.
 - [31] M. Boardman and T. Trappenberg, "A heuristic for free parameter optimization with support vector machines," in *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, Vancouver, BC, 2006.
 - [32] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5-23, 2001.
 - [33] E. Ukkonen, "On-line construction of suffix trees," *Algorithmica*, vol. 14, no. 3, pp. 249-260, 1995.
 - [34] U. Manber and G. Myers, "Suffix arrays: A new method for on-line string searches," *SIAM Journal on Computing*, vol. 22, no. 5, p. 935–948, 1992.

- [35] C. Wang, H. Wang, Z. Xie and J. Gao, "Exact string matching with variable length of don't cares based on suffix tree," in *International Conference on Information Science and Technology (ICIST)*, Nanjing, 2011.
- [36] M. Barsky, U. Stege, A. Thomo and C. Upton, "A new method for indexing genomes using on-disk suffix trees," in *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, New York, 2008.
- [37] B. A. Lewis, R. R. Walia, M. Terribilini, J. Ferguson, C. Zheng, V. Honavar and D. Dobbs, "PRIDB: a protein–RNA interface database," *Nucleic Acids Research*, vol. 39, no. 1, pp. 277-282, 2010.
- [38] E. Frank, M. Hall, L. Trigg, G. Holmes and I. H. Witten, "Data mining in bioinformatics using Weka," *Bioinformatics*, vol. 20, no. 15, pp. 2479-2481, 2004.
- [39] C. Chothia and A. M. Lesk, "The relation between the divergence of sequence and structure in proteins," *The EMBO Journal*, vol. 5, no. 4, pp. 823-826, 1986.
- [40] M. Zuker, "Mfold web server for nucleic acid folding and hybridization prediction," *Nucleic Acids Research*, vol. 31, no. 13, pp. 3406-3415, 2003.
- [41] E. Ukkonen, "On-line construction of suffix trees," *Algorithmica*, vol. 14, no. 3, pp. 249-260, 1995.
- [42] "Encoding Suffix Trees Solved by 210," [Online]. Available: <http://rosalind.info/problems/suff/>. [Accessed 24 January 2016].
- [43] M. Elias, D. Liebschner, J. Koepke, C. Lecomte, B. Guillot, C. Jelsch and E. Chabriere, "Hydrogen atoms in protein structures: high-resolution X-ray diffraction structure of the DFPase," *BMC Research Notes*, vol. 6, p. 308, 2013.
- [44] B. Tian, P. C. Bevilacqua, A. Diegelman-Parente and M. B. Mathews, "The double-stranded-RNA-binding motif: Interference and much more," *Nature Reviews Molecular Cell Biology*, vol. 5, no. 12, pp. 1013-1023, 2004.