APTAMER CLASSIFICATION THROUGH NOVEL PROCESSING

OF NUCLEOTIDE SEQUENCES

Approved By:

_____

Dr. Margaret Dunham

_____

Dr. Francis Coyle

_____

Dr. Michael Hahsler

APTAMER CLASSIFICATION THROUGH NOVEL PROCESSING OF NUCLEOTIDE SEQUENCES

A Thesis Presented to the Undergraduate Faculty of

Southern Methodist University

in

Partial Fulfillment of the Requirements

for the degree of

Bacherlor of Science

with a

Major in Computer Science

by

Vladimir Jovanovic

May 12, 2012

Jovanovic, Vladimir

Aptamer Classification through Novel Processing

Of Nucleotide Sequences

Advisor: Professor Margaret Dunham

Bacherlor of Science conferred May 12, 2012

Thesis completed May 8, 2012

Aptamers are short strands of DNA or RNA that have a high affinity towards bonding with certain molecules. Strands of DNA can fold on to themselves to form rudimentary shapes of stems and circles called the secondary structure. The secondary structure of single-stranded nucleotide sequences (DNA and RNA strands) is necessary to allow the binding for aptamers. From this, a thesis is presented that a summarization of all the possible secondary structures that a strand can form is pertinent to aptamer classification. Two novel methods based off of this idea are created, dubbed Palindrome Counter and Belt Counter.

The two counter methods are tested against a Naïve method of representing the DNA sequences numerically base by base. Two experiments are conducted to test the hypothesis that the counter methods would do better than chance or the Naïve method across different classification models. The results show that there was no significant difference between the three methods, but Belt Counter was better at classifying non-aptamers than the Naïve method giving some support to the hypothesis.

TABLE OF CONTENTS

LIST OF ALGORITHMS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEGEMENS

I would like to thank Dr. Douglas Raiford and Dr. Stephen Lodmell from The University of Montana for providing the aptamer data.

Chapter 1

INTRODUCTION

Aptamers are short strands of deoxyribonucleic acids (DNA) that are usually 30 to 50 nucleotides long [1]. To be defined as an aptamer, the short DNA strand has to be able to bind to, or have a high affinity towards, other specific molecules. These molecules can range from large structures like cells, to smaller scale forms like organic compounds, or proteins [2]. Given an aptamer, the strand by definition should be able to bind to one particular molecular structure with relative ease. The determination of which strands can be classified as aptamers and thus bind to a certain structure is the discussion of this thesis. Novel classification methods will be covered along with the data used to create them, the classification efficacy, and potential future uses.

1.1 APTAMER STRUCTURE AND BASICS

There are three primary pieces to any aptamer strand when viewing it from a one dimensional perspective (the sequence of letters). There is a header and tail sequence (primer overhang sequences) of usually the same nucleotide pattern followed by the middle with the remainder of the nucleotides. The primer overhangs are used to allow easy cloning of the DNA sequences and are discussed in section 1.2. They are the same for all the artificial aptamers that are made in a lab [3]. While DNA is usually used by an organism to create a new protein, aptamer sequences are not. Even though they can be

formed with ribonucleic acids (RNA), aptamers are not used to directly translate to protein sequences. The structure of RNA sequences that translate to protein is different from aptamers. It usually includes a long poly-A tail, multiple repetitions of the nucleotide adenine. Thus, due to not having to formulate proteins, RNA versions of aptamers do not look like protein coding RNA strands.

The aptamers bind to target molecules by the secondary structures they can create through hydrogen bonds across complementary nucleotides. Each DNA strand has long chain of a four possible bases: (A)denine, (T)hymine, (C)ytosine, (G)uanine. As the parenthesized capital letters indicate, they are represented by four letters: A, T, C, and G. The molecular shape of A and C allow them to pair up easily with T and G, respectively. This pairing is due to hydrogen bonds that complement the pair of nucleotides to hold them together. Hydrogen bonds are not permanent though and can be easily broken and remade with heat. RNA, as mentioned earlier, is very similar to DNA except for the use of (U)rasil instead of thymine.

When a DNA or RNA strand is left in an aqueous solution with no complement, the strand contorts and folds over itself. The hydrogen bonds work across bases in the same strand and thus certain structures can be formed. This folding across the strand creates the secondary structure of the DNA/RNA strand. The primary structure is the list of nucleotides from head to tail. The secondary structure of these strands have stems of paired up bases, circles where no hydrogen binding occurs, and overhanging end bases that did not pair up to any particular nucleotide. Each unique strand, and even the same sequence given another chance, will produce a different secondary structure. This structure is what allows an aptamer to bind to a particular molecule. Figure 1 is a sample

aptamer that is in the data discussed in the next chapter. Here, the basic structure of the folded RNA strand can be seen. In particular, there are small circles of non-binding that occur followed by almost ladder-like hydrogen bonds between complementary pairs of nucleotides (C-G and A-U). Any single strand has many possible folding shapes it can make, and sophisticated algorithms are required to determine which are most likely to happen.

Figure 1 was created through one of the most famous algorithms developed in 1980. This algorithm calculates the lowest possible free energy for the strand by looking at all the possible stems and loops the strand in particular can make [4] Each loop has bases exposed to the environment that allow the strand to shift and change shape. There is a high free energy associated with exposed bases as opposed to those that have already complemented to another base. By finding the combination of loops that has the lowest amount of free energy, that complete structure is most likely to form [4]. Of course, this method only takes into account sequences at 37^{o} Celsius, and thus is more of a heuristic. If the temperature were to change, the structure can change due to another orientation having less free energy.

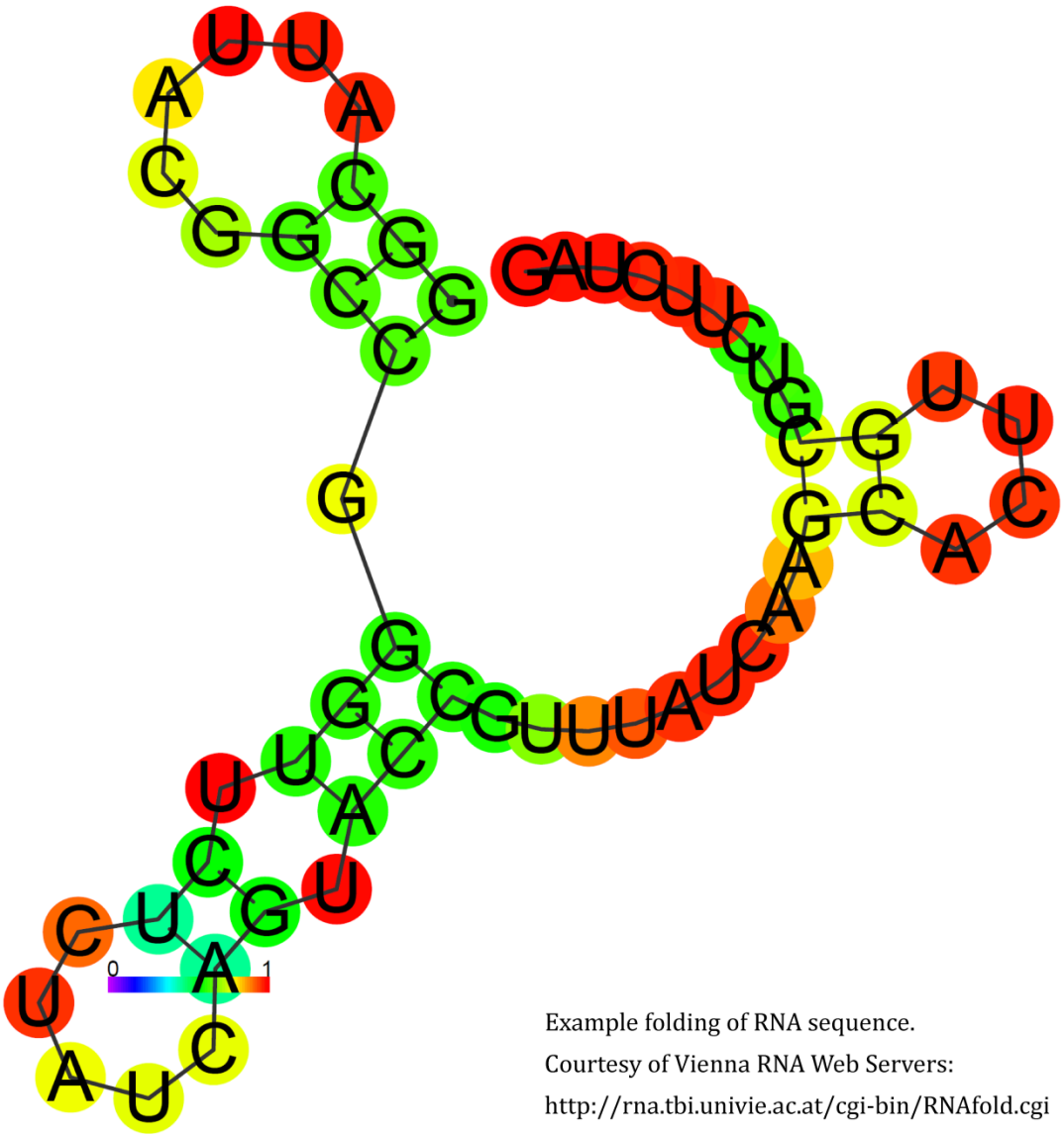

Figure 1. RNA sequence with folding in secondary structure form.

1.2 APTAMER CONSTRUCTION

Aptamers were first created in a laboratory setting before their discovery in cells. Using a method called SELEX, "often called in vitro selection or [evolution]," scientists were able to create DNA or RNA (ribonucleic acid) sequences that have a high-affinity for a certain protein [5]. A large pool of random DNA sequences is created with random

lengths of nucleotides. Each sequence has the same primer overhang in a batch. As mentioned before, these allow the strands to be replicated easily. The molecule that the strands are supposed to bind to is introduced. Through fluorescent tagging, researchers are able to choose only the strands that are able to bind to the molecule for further purification. The strands that are chosen are then amplified through polymerase chain reaction (PCR) [5].

PCR is a method of amplification by taking a DNA strand and making copies of it repeatedly. Each produced copy allows double the replication (as there is twice the strands to copy from before) thus allowing exponential replication of the small amount of strands that can bind to the molecule. The strands need the primer overhang sequence on the head and tail for PCR. To replicate the strands, each DNA strand needs to have a starting amount of DNA strand attached at the head and tail of the DNA sequence. By knowing what the primer overhang sequences are for both sides, lab technicians can introduce complementary primer sequences. These will bind to their complementing primary overhang sequences through hydrogen bonding. The protein polymerase can then create the full complement DNA strand, and from there, create the original DNA copies [6]. Special care needs to be taken in choosing the primers as some can negatively affect the PCR process by not correctly binding to the primer overhang heads and tails [5]. An example would be a long single nucleotide chain of repeating A's as the primer. While this will bind to the complement of a chain of T's, the result could be improperly aligned. One of the middle nucleotides could misalign and form a bump due to nucleotides further down the sequence attaching to its complementing base. These little bumps or kinks prevent proper copying of the DNA and reduce the efficiency of the whole process [5].

The strands that were chosen and replicated are introduced to the target molecule from before. Only this time, a stricter threshold is used to detect which sequences bind to the molecule [5]. These more easily binding strands go through PCR again, and the method of selection is repeated with stricter thresholds. This cycle is repeated several times to have pure batches of aptamers that bind to the particular molecule that all have the same primers. What are left are sequences of DNA that have a high affinity towards a certain molecule. In overview, from a starting sample of 10^6 randomly generated sequences, SELEX creates a batch of aptamers that have a high affinity for binding to one particular molecule in "a matter of days" [7].

Other methods of aptamer creation do exist though as evidenced by Closed Loop Aptameric Directed Evolution (CLADE for short) [8]. This is a very similar method to SELEX except that it can be computer simulated, or be *in silico* (in computer). While the two methods outlined create artificial aptamers, natural selection in biology has found ways to create aptamers. MicroRNA (miRNA for short) is a short 20 nucleotide long strand that binds to particular strands of other RNA that code for certain proteins, thus regulating the expression of that protein [9]. These strands of RNA are encoded in the DNA in the nucleus of an organism. As soon as the sequence of DNA that codes for RNA is replicated, the resulting miRNA strand leaves the nucleus and binds to its target.

In fact, this regulation of protein expression is one of the key uses of aptamers in biology. Each miRNA strand has a certain capability of binding to a piece of RNA strand that code for a protein (usually referred to as mRNA). Depending on the specific sequence of the miRNA, the protein coding mRNA strand could end up stopping at the right place for the protein synthesis or be abruptly cut off [9]. This process can create a

non-functioning protein which can have wide spread effects in an organism and affect the development of many key systems.

1.3 USES

In the laboratory setting, the aptamers created have a great use as well. Depending on the nucleotide bases used in the DNA synthesis, the aptamers can be made to glow under fluorescent lighting. As mentioned before, this is how researchers can determine which strands are aptamers for a specific molecule. Once a pure aptamer strain is developed, it can be used to easily tag certain parts of an organism to allow easy identification of key structures. This method of biotagging has already been implemented to create a strain of aptamers that can bind to HPV-associated cervical cancer cells [10]. By taking a sample of cervical tissue from a patient, doctors could coat the sample in the aptamer to see where the strands bind to identify if there are cancerous cells and what proportion of the sample are cancerous.

The use of aptamers do not have to be relegated to just biotagging. As mentioned before, the naturally occurring aptamers like miRNA are used by cells to regulate the production of proteins. Artificial aptamers created through SELEX could do the same job as miRNA. Even more so, they could attach to parts of DNA inside the nucleus to have much wider control of gene expression rather than at the individual protein level. Research has already been done looking into the use of artificial aptamers for this [11].

1.4 OVERVIEW OF THESIS

Aptamers have the potential to be used in a wide variety of biological uses as shown in the previous section. And as mentioned earlier, it can take days to get a purified strain of aptamers to be able to bind to a particular molecule. Unfortunately,

SELEX is limited in that aptamers can have an affinity towards similar molecules as the targets they are intended for. Because of this, creating batches of aptamers that can bind to only one molecule as opposed to any other can take a much longer time. By the sheer fact that aptamer selection begins with a random assortment of over a million different DNA strands, data mining is the natural next step in aptamer research. Using data mining, creating a strain of aptamers can become more efficient as new classification methods can be tuned to finding strains that attach to only one molecule. And then, aptamers can be compared against one another to see if they would bind to similar molecules using these same classification methods. Thus, classification of aptamers through data mining has the potential to increase the efficiency of aptamer creation and help it target more specific targets without interfering with other ones.

1.4.1 Previous Research

The easiest place to start the research for aptamers is to look into miRNA. Due to the large size of human DNA, it is not possible to test every region of DNA to see if a particular segment is miRNA. Instead, using data mining, models can be created to predict if certain strands of RNA are miRNA or something else. It is even possible now to go online and see if a strand could be miRNA through a program called MiPred [12]. This program uses the Random Forest model to classify miRNA. Another classification method dubbed "miRFam" classifies miRNA according to subclasses of miRNA that share similar structures using only the sequence information [13].

Research is also being conducted into aptamer classification as well. People have investigated combining SELEX with yeast DNA to use the yeast protein manufacture as a way to carry out aptamer classification [14]. Other researchers took a look at the primary

and secondary structure of aptamers to first group them across similar features to see if there was any particular pattern in their binding [15]. They found that secondary structure was determined by fixed sequences in the aptamers.

Besides the previous research, it was difficult to find other published research on aptamer classification. I was not able to find a methodology to use data mining to create a model to classify between aptamers that bind to a target and non-aptamers that do not bind to the target. My thesis and this research then fill in an important gap in the knowledge. Whether the gap is caused by the obscurity of the previous research or lack of research is unknown.

1.4.2 Thesis

As evidenced in previous research and sections, the secondary structure of a strand of DNA or RNA determines if it will bind to a specific target. If the strand does bind to the target, then it is an aptamer. Classification between aptamers and non-aptamers would therefore need to take into account the secondary structure of the DNA/RNA strands. The problem is that determining what exact secondary structure is formed by a particular strand is NP-complete [16]. While it is possible determine structures given certain constraints [4], the results are not necessarily accurate given the same strand in all scenarios. Thus, analyzing the true secondary structure of each potential aptamer to classify it would be inevitably complex and fruitless compared to the previous method of using SELEX.

I propose that looking at the primary structure of nucleotides can give a glimpse into the secondary structure in order to determine the classification. I further propose that, for a given set of aptamers that all bind to a target, they do it through multiple ways.

Instead of all binding at the same site on a particular molecule, there could be multiple binding sites for different orientations of the multiple strands. Thus, a method to summarize all of the potential secondary structures would be of more benefit for aptamer classification. I also propose from this, by looking at the possible self-pairings of nucleotides for a given strand, that a pattern can emerge to help in classifying aptamers. In this case, I assume that a given strand can fold in multiple ways depending on how many nucleotides pair up. If none of the nucleotides pair with each other, then the strand will not fold over itself. However, looking at the different ways the aptamer can pair the bases should lead to a glimpse into the possible secondary structures it can form. Every location that has pairings would form a stem, while the non-complementary bases would form circles.

1.4.3 Hypothesis

Given my theory of how aptamer classification could work, I hypothesize that:

(1) A preprocessing method of a group of DNA strands that lists all of the possible base pairings for each strand would lead to an effective classification between aptamers and non-aptamers.

(2) The preprocessing method should boost the performance of any classification model used.

For the purpose of my hypotheses, effective is defined as performing better than chance or a naïve implementation of aptamer classification. Further, aptamers are defined as DNA/RNA strands that bind to a particular target, and non-aptamers are defined as those strands that do not bind to that target. Finally, a naïve implementation would simply look at the nucleotide sequence to classify between aptamer and non-aptamer.

If support is given for this hypothesis through my testing, then that will give credence to my thesis of how secondary structures influence aptamer binding. My method of classification could also lead the way into potential new ground for research in aptamer construction based on the thesis.

There will be two experiments conducted. The first will test three different methods of transforming DNA sequence data. Two will be based on my thesis, and the third will be the naïve method. Training models will be made against all three data sets and the efficacy between them will be tested. The next experiment will take the best performing model from the first experiment to see how the transforming methods due against another set of data.

1.4.4 Paper Organization

The next sections of the paper will cover the testing of my hypothesis. My hypothesis will be tested by covering two methods I developed to incorporate my thesis. The paper will follow the knowledge discovery in databases process [17]. The initial data will be disclosed with how it was selected. Then the preprocessing methods will be covered. My two methods and the naïve approach to classification will be covered with the data transformation step. Finally, the actual data mining applied to the data and the subsequent interpretation of the data is discussed. Figure 2 illustrates my methodology and where my thesis fits into a data mining approach to classifying between aptamers and non-aptamers.

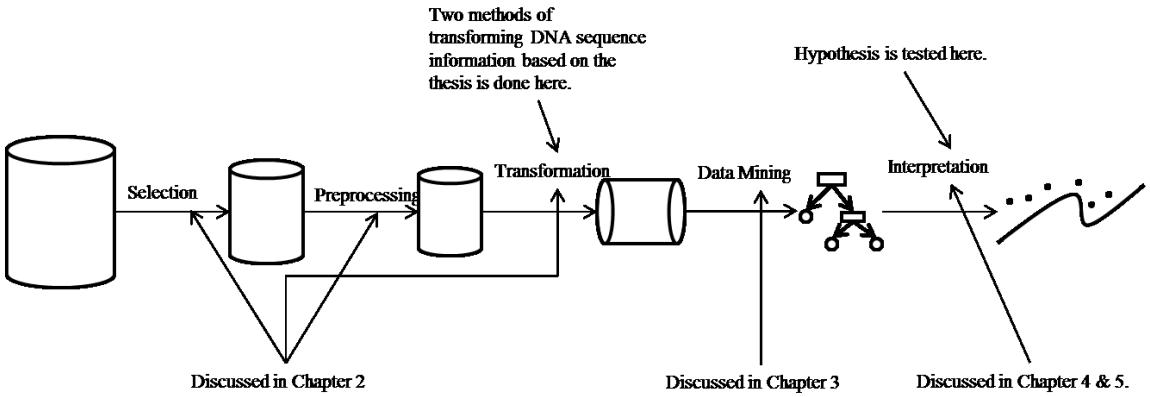

Figure 2. Knowledge Discovery Process modified from [17]. This outlines which chapters pertain to key steps in data mining.

Chapter 2

DATA MANIPULATION

Aptamer data consists of a list of DNA or RNA sequences that have been shown to have a high affinity to a particular molecule. In the case of testing my hypothesis, aptamer data was used that had a high affinity to binding towards the Rift Valley Fever Virus. This data was graciously provided by Dr. Douglas Raiford and Dr. Stephen Lodmell at the University of Montana. They provided multiple specimens of aptamers that binded to the virus along with some non-aptamers, sequences that were shown to not bind to the virus.

2.1 SELECTION

The first batch of data that was provided was a 126 aptamer/non-aptamer file. It contained the sequences followed by the classification of aptamer and non-aptamer. Sixty three were non-aptamers while 54 were aptamers that binded towards the virus. This data was further supplemented by a list of 100 aptamers that binded to the virus as well. Unfortunately, some of these were repeated in the original file thus leaving a sample of 39 aptamer strands. The primers were identical on both data sets of aptamers with headers 14 nucleotides long and the tails at 12 nucleotides.

The non-aptamers provided were created by using a random number generator while keeping the primer and tail consistent. This would be fine, except for the small concern that some of the randomly produced strings could have a chance of having a high affinity to the virus were it to be created into an actual DNA strand. This is further discussed in Section 5.1 Limitations.

2.1.1 Sequence Alignment

One issue brought before is how many conserved regions are there across aptamers. In other words, is it possible that aptamer classification depends simply on knowing what the sequences of nucleotides are? If this is the case, then multiple sequence alignment should be able to show if there are sections of genetic code that are shared across all the sequences. Certainly, the primer overhangs on the head and tail will have an alignment across all the aptamers.

Appendix A shows the results of the alignment using ClustalW2 as the sequence alignment algorithm [18]. The data used was the second file of 100 aptamers. Because this contained all of the sequences to be used in the testing, it only made sense to use this complete file. As predicted, there were matches along the primer overhangs. There were also two large clusters of conserved region that are roughly 14 and 10 bases long. There was also a conserved region of about four bases, but it was not seen through all of the sequences. This does bring cause for concern to my thesis as this basically assumes that the primary structure of a DNA strand will determine aptamers even though this is contrary to previous data. It is possible that there are other unlisted aptamers that were never listed due to one reason or another though. And, to account for this possible simple

view of aptamer classification, the naïve method of transforming the data should cover this.

2.1.2 Levenshtein Distance

The number of edits required to convert one string to another is the Levenshtein Distance algorithm. This can be useful in biology when trying to determine how similar or dissimilar the DNA sequences are. In the case of the aptamers given, I chose two pairs that had the lowest score and highest score as given by [18]. The score is calculated as the percentage of the same bases in the same position as the other string. What is interesting to note though is that there were multiple pairings in the 100 piece data that had a score of 100. Thus the distance would be calculated to be 0. In other words, there were multiple aptamers with the same base sequence. For the lowest performing pair, their score was calculated at 58 while the Levensthein Distance was calculated at 21 [19].

2.2 PREPROCESSING

The data, while not necessarily abundant, still needed to be cut. Many of the aptamers listed had variable lengths of nucleotide bases due to the random mutations in the SELEX creation of the aptamers. My methods of transformation unfortunately require consistent length across all DNA sequences. I had to cut out all the aptamers that did not have the same length as the non-aptamers. This resulted in aptamers and non-aptamers that were only 56 nucleotides long. In the end, the first batch had 50 aptamers and 63 non-aptamers left. The second batch had 32 left.

2.3 TRANSFORMATION

As was mentioned before, the secondary structure of the DNA and RNA strands are what allow them to become aptamers to particular molecules. I present here two

novel methods of summarizing the secondary structure of the strands. On top of this, I will cover the naïve method to be tested against these two in the experiments. First, it is important to remember that what helps define a secondary structure is the amount of pairings that the nucleotide bases can have against one another. In Figure 1, there is a large circle in the center primarily due to the lack of ability for many of the bases to pair up with their complements. My methods take this into account by counting how many possible complementary pairs are made at each possible position of an aptamer. In essence, my two methods assume that by folding the strand over in half at each possible position, the total count of base pairings can help define how the structure behaves.

The more base pairings there are, the longer and thinner the strand would eventually turn into. Conversely, the less base pairings there were in positions, the more bulbous would the strands finally look like. By capturing this simple description of the strands, a classification algorithm can be used to create rules or find associations between the counts and what defines an aptamer. By showing enough sequences of numbers for the aptamers that bind to the virus, there should be a classification method that could easily classify new strands as aptamer or not simply by the transformation methods I have.

Transformation method Palindrome Counter and Naïve were both developed in an earlier project of mine. These were two possible ways of summarizing DNA sequences for aptamer classification that I thought up. Belt Counter is an extension of Palindrome Counter [20].

2.3.1 Palindrome Counter

The first transformation method is called the Palindrome Counter. To give a visual perspective, the program takes a given nucleotide sequence. It grabs the far most left base and pulls it along the top of the remainder of the strand until the whole order has been reversed. By taking a count of the base pairings at every move of the top strand, a vector, or list of numbers, can be stored. These can represent how many bases pair for every shift of the DNA strand. Figure 3 demonstrates this method as it was described.

Figure 3. Palindrome Counter shown in visual form.

The full Palindrome Counter program is detailed in Appendix B. Listed here is a pseudo-code description of the Palindrome Counter algorithm as Algorithm 1.

```
Input:
   DNA // List of DNA strings
Output:
   V // Set of vectors of complementary counts
Palindrome Counter Algorithm:
   for each d ∈ DNA do  // d is a string of letters
      C = {}; // Empty vector for counts
      for i to lengthOf(d)-1 do
         count = 0
         j = i+1;
         k = i;
         while k >= 0 and j < lengthOf(d) do
            if d.at(k) is complement of d.at(j) then
               count++;
            k--;
            j++;
         C = C ∪ count; // Add the number of base pairs
      V = V ∪ C;
```

Algorithm 1. Palindrome Counter pseudo-code to convert a DNA strand into a list of complementary counts.

2.3.2 Belt Counter

The second method that will be tested in efficacy alongside Palindrome Counter is the Belt Counter. This method is very similar to Palindrome Counter except that the two ends of the single strand are connected and then any pairings across the "belt" formed are counted. The belt buckle, the junction of the two ends, is allowed to shift from left to right to get all the possible combinations of the two end pieces against the other side. In essence, this is equivalent to Palindrome Counter, except that the tail end, instead of it being left dangling, is attached to the beginning of the strand as it moves across. Figure 4 demonstrates a visual representation of Belt Counter.

Figure 4. Belt Counter in visual form. The ends of the strand are signified by the straight line. As the program goes through the sequence, the end of the strand shifts from left to right.

The data represented by Belt Counter is actually a summarization of Palindrome Counter. While Palindrome Counter would have a vector of length 55, Belt Counter has a vector half the length of that (anymore and the loop structure would repeat itself). The data for the Belt Counter, for the parts after the "buckle" are actually the last section of the equivalent Palindrome Counter data but in reverse. Basically, the data is accounted for in half of the amount numbers required for Palindrome Counter. Appendix C details the full Java code for Belt Counter, and Algorithm 2 is a pseudo code version of it.

```
Input:
   DNA  // List of DNA strings
Output:
   V  // Set of vectors of complementary counts
Belt Counter Algorithm:
   for each d ∈ DNA do  // d is a string of letters
      C = {}; // Empty vector for counts
      for i to lengthOf(d)/2+1 do
      count = 0
         j = i+1;
      k = i;
         while k >= 0 and j < lengthOf(d) do
            if d.at(k) is complement of d.at(j) then
               count++;
            k--;
            j++;
         k = lengthOf(d)-1;  // Count pairs from tail
         while k != j and j < lengthOf(d) do
            if d.at(k) is complement of d.at(j) then
               count++;
            k--;
            j++;
         C = C ∪ count;  // Add the number of base pairs
      V = V ∪ C;
```

Algorithm 2. Belt Counter pseudo-code to convert a DNA strand into a list of complementary counts.

2.3.3 Naïve Method

The Naïve method is simply a translation of the DNA values into numbers. The bases A,T,C,G correspond to 1,2,3,4 [20]. This method will be used to test whether the positions of specific bases help determine the classification of the aptamers.

2.3.4 Data Creation

Appendix B and C show the code in Java that I used to scan in the aptamer sequences to make these lists of numbers. They go with Palindrome Counter and Belt Counter respectively. Both programs have a complexity of $O(n^2)$. This method of

secondary structure analysis is actually faster than the one that formed Figure 1 with a complexity of $O(n^3)$ [4]. Each of the aptamers and non-aptamers were converted into these two vector formats and kept in separate files. To have a proper comparison of training and test though, there needed to be some extra data. I created a random DNA sequencer outlined in Appendix D to create a list of random nucleotides at a user-inputted length. This was attached to the second set of all aptamer data to give a proper comparison of the test data. Without the non-aptamers thrown in, assuming they would all actually work out to be non-aptamers *in vivo* (in living cells or nature), the model would only be testing the efficacy of any model for one half of the classification. Finally, the raw DNA base data for both training and test was converted to numbered values as mentioned for the Naïve Method.

Chapter 3

DATA MINING

With all the data prepared, there were six files in total. A training and test set for the Palindrome Counter, Belt Counter, and Naïve Method. The training set had 50 aptamers and 63 non-aptamers. The testing set had 32 aptamers and 32 non-aptamers. The full training set is used to create the model, while the testing set is used to test to see the efficacy of the model. The primary goal of this research is to show how the new transformation methods can be used to aide in classifying aptamers, so there will not be an incisive critique of which classification methods are the best. Instead, the Experimenter function of Weka will be used to explore how effective different classification methods are in determining if a given strand is an aptamer or non-aptamer.

Weka is a free Java based data mining tool [21]. It has a multiple implementations of the most common classification algorithms including functional and decision tree. This tool was used due to its ease of use. Also, its experimenter function allows a user to easily test multiple models side by side to see their performance. There are many variables that can be adjusted for each of the models, unfortunately testing all the possible combinations of the parameters for each model along with the model itself would take too long and would become unnecessarily complex. My tests with Random Forest and Dagging had shown that the model can achieve close to 100% accuracy in

classifying a given aptamer strand as an aptamer [22]. No parameters were changed in these models, so it's safe to assume that other models would work comparably well.

To use Weka, the first thing I had to do was to create .arff files of all the data sets that I will be using. Unfortunately, while .csv files are supported in Weka, manipulation of the data pre- and posttest is much easier if the file is in .arff format. With this done though, I began to test out the same set of models against the Palindrome Count, the Belt Count, and Naïve Method. In the next sections, the performance of Palindrome Count, Belt Count, and Naïve Method will be discussed separately as far as model training with cross-validation. Comparisons will be made later on.

3.1 TRAINING EXPERIMENT

In the experimenter, the models that were used were Simple Logistic, Decision Table, J48, Random Forest, Naïve Bayes, Multilayer Perceptron, and Dagging. Each experiment went through a cross-validation ten times along with having the experiment performed ten times. Cross-validation takes a given data set and separates between training and test data depending on how many "folds" are set for the validation method. In the case of my test, I chose the standard 10 fold cross-validation which means the data was split 90% training and 10% testing. Thus all ten groups of possible testing and training were tested against each other to see their performance. Finally, each of these experiments is run ten times to see how changing the starting seed value affects the results of the model training. The need for this repetition is to have different training groups to have more significant results (by having more experiments run).

3.1.1 Model Setup

Decision trees to classify DNA sequences have been established as effective models [20]. In this light, I chose Random Forest and J48 to test out if this was the case with my data. The default parameter settings were used for both trees. I based this on my previous research on this topic [20]. I chose J48 because it is a decision tree that Random Forest does not use. To have a fair representation of decision trees then, I had to include both.

Decision Table was chosen to test out how a rule based model would work with the data. Because the attributes, or numbers, in the data do not follow any necessary logical order or affect each other in a logical manner, Decision Table is much like a null hypothesis test. It should do the worst comparatively because it expects features to affect one another to do a classification. Again, the parameters were all left at default for this method.

Simple Logistic is Weka's implementation of a linear logistic regression model. This method was chosen because it is specifically used to classify between two classes. Considering the data is either aptamer or non-aptamer, it makes sense to use it. The default values were used for this method.

Naïve Bayes was chosen because it assumes all the features in a data set are independent of each other. I figured that this would be a good test to see how well the three methods perform under this assumption. This would be exactly opposite to Decision Table which assumes that the attributes are not independent of one another. Here, there are no parameters to be set besides the debugging setting. Everything was left as default.

Multilayer Perceptron was chosen because a neural network is one of the quintessential data mining algorithms. The model is a three layer neural network. I preferred to not set the model each time and let the neural net build itself. Thus, the default settings were used for this as well.

Finally, Dagging was used because by chance, it performed well in previous research on this data [22]. Dagging is a majority vote system of a couple of base classifiers. In other words, the user can select which classifiers to use, and then Dagging will take a vote from all the classifiers selected to determine what the classification is [23]. I used the default setting in my previous research and use it here as well. I am keeping this method here in order to see if my previous results were spurious or not.

3.1.2 Results

The results of just purely looking at the raw transformed data gives an accuracy of about 90% across all of the experiments run 100 times each. By looking into the results file, I was able to see that many times, the percent correct in classification was 100%, mixed in with other cases of only 70%. This would be the case in cross-validation when the training and test size can make a difference. The one model that did do the most poorly was the Decision Table at 76.13% correct for Palindrome Counter, 80.44% correct for Belt Counter, and 85.53% for Naïve Method across all the tests. This makes sense as a Decision Table would not intuitively work with data that is not dependent on each other. The data is represented visually in Figure 3. The results copied directly from the Experimenter evaluations are in Appendix E.

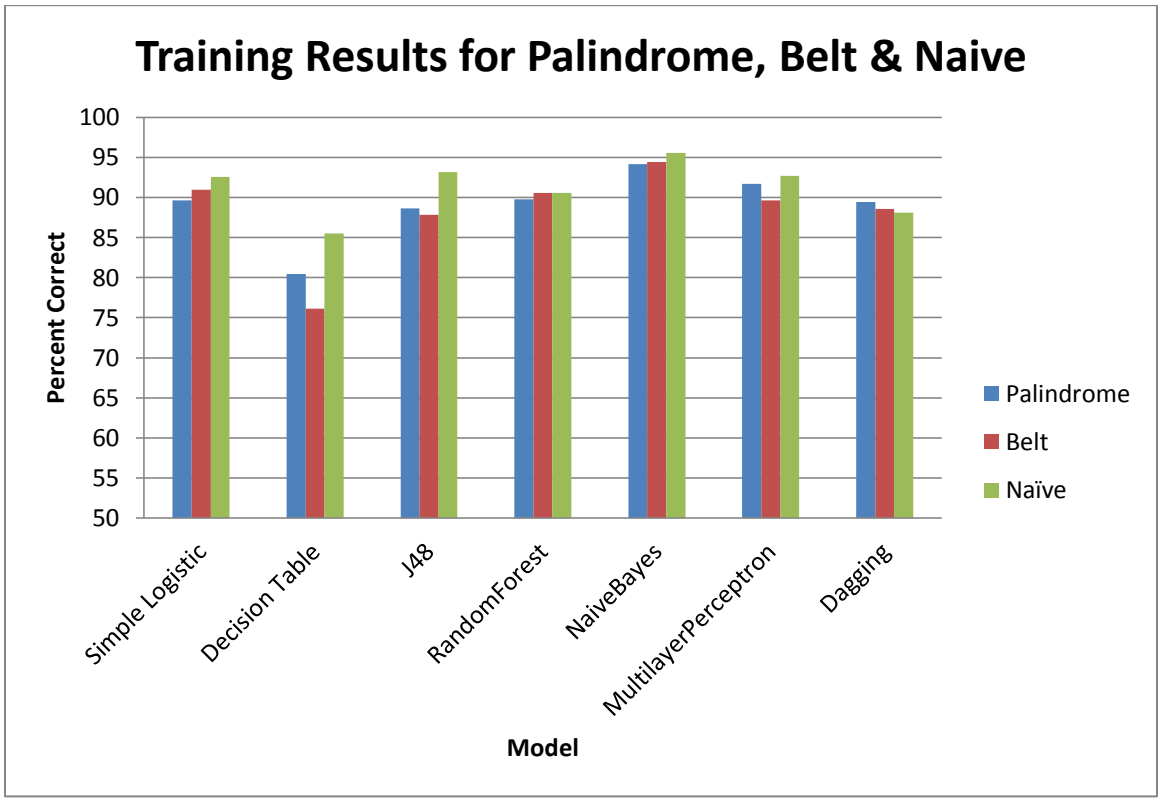

Figure 5. Bar graph comparing how the two Counter methods and Naive did in the Experiment phase.

Another interesting thing to note is that the Decision Table actually did significantly worse when compared to the other models. By significantly worse, Weka Experimenter is able to do significance testing based on the percent correct for all possible pair-wise combinations. Weka uses the paired T-test to find the significance of the difference in values between two given sets. In the case of Naïve Method versus the two Counter methods, there was not a significance difference. This means that the significance value was greater than .05, or that the probability of the two sets of values being the same rather than different was greater than 5%. While the averages were

obviously different in terms of percent correctly classified, all other models had no significant difference between each other. In fact, they all did significantly better than the Decision Table except for Random Forest with regards to the Palindrome Counter data. There, there was no significant difference in the performance of Decision Table compared to Random Forest with Belt Counter data.

While these results are promising, it is always a good idea to check to see if there are ways to improve the accuracy and efficacy of these models. I did this by using CfsSubsetEval in the Explorer window of Weka. The subset evaluator takes all the possible subsets for the attributes and subtracts those that are highly correlated. In essence, this reduces the redundancy of the information stored in the .arff files. Methods like Naïve Bayes are able to improve because it assumes that all attributes are independent and not correlated to one another. By choosing the training data for Palindrome, the algorithm determines which attributes are needed to differentiate between the two classes. In the end, 16 of the 55 numbers were kept. Doing the same with Belt Counter, CfsSubsetEval determined that only 12 attributes of the 29 were needed. The same attributes were removed from the testing data as well and stored as new files. Finally, the Naïve method had only nine attributes of the 56 originally. This indicates, that the Naïve method was able to benefit greatly from the subset evaluator by taking out much needless information for classification.

The same experiment was run except with the new data. The results show that using the attribute filter, the scores did improve somewhat for some of the models. The highest percent correct was with Naïve Bayes going as high as 97% in Palindrome

Counter, 95% for Belt Counter, and 95.37% for Naïve Method. Figure 4 elaborates these differences further, along with a list of the results from the Experimenter in Appendix F.

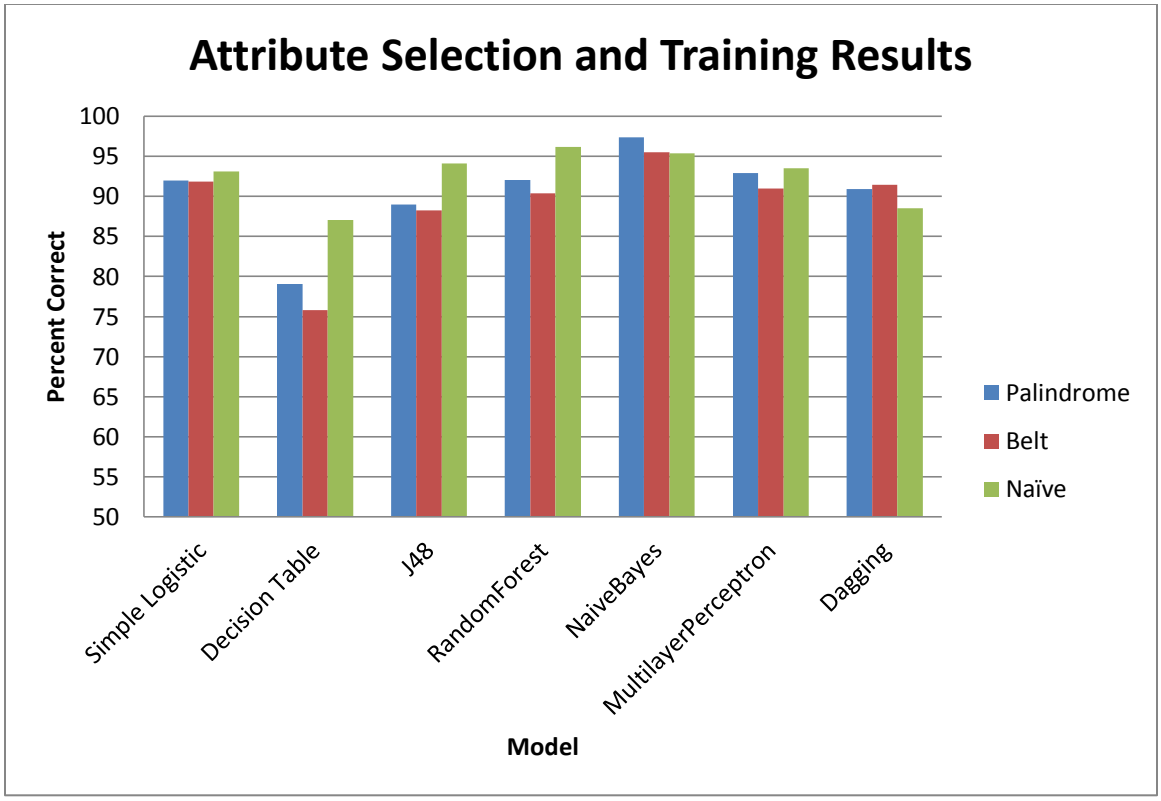

Figure 6. The effect of attribute selection on the experimenter results.

Yet again, Decision Table is the worst performing model according to percent correctly classified. In every case, the other models did significantly better than Decision Table in percent correct. Naïve Bayes is clearly seen as the best performing. It did significantly better in the Palindrome Counter tests against models: Simple Logistic, Dagging, and J48. It did not perform significantly better with Belt Counter in any other model.

3.2 TESTING EXPERIMENT

From the results of the Experimenter phase, it is clear that Naïve Bayes is the best performing model. Ranking the number of significant differences among the models by data, Naïve Bayes had more occurrences of being significantly different than the other models. Therefore, it will be used in the Explorer function to create the final model to test against the second set of data. This will prevent needless repetition of results. As Naïve Bayes was significantly better across multiple tests against multiple models, it makes sense to use it as the basis of our model.

To use the testing data, the full training data is used to create the model. Then, the testing data with the randomly generated DNA strands and other aptamers is input. Weka calculates a confusion matrix and determines the necessary statistics. This was done for Palindrome Counter, Belt Counter, and Naïve Method for both the full data and selected attribute data. The comparison between the two methods will be discussed fully in the next chapter.

3.2.1 Palindrome Testing

Palindrome Counter had some good results before when I first ran the program in preliminary testing. I was expecting to have a high accuracy, as the aptamer only classification from before had results as high as 97% [22]. Unfortunately, those preliminary findings did not show the whole story about the method. Due to their being no non-aptamers in the training set, the initial results had a skewed precision rate. While all of the aptamers were for the most part recalled correctly, there was no way to properly test the precision, or how many of the classified aptamers were actually aptamers.

In the first test with all the attributes, Palindrome Counter was able to correctly classify 64% of the data. In the second test, 75% of the data was correctly classified. Table 1 displays the confusion matrix for both of these instances.

	Full Attributes		Selected Attributes	
Classified As ->	Aptamer	Non-Aptamer	Aptamer	Non-Aptamer
Aptamer	29	3	30	2
Non-Aptamer	20	12	14	18
Precision	0.592	0.8	0.682	0.9
Recall	0.906	0.375	0.938	0.563

Table 1. Confusion matrix comparing Palindrome Counter results with precision and recall divided by classification.

Looking at the confusion matrix and precision and recall rates, full attributes has a high recall rate for aptamers and a high precision for non-aptamers. Basically, when given a strand that actually is an aptamer, this current model has a good chance of classifying it correctly as an aptamer – not so with the case of non-aptamers. The odds of being correct once something is identified as non-aptamer are quite high. In this sense, its use of the non-aptamer classification is more precise than the aptamer one. The specific attributes selection shows that there is an overall increase in the rates for both precision and recall. The main difference in this model compared to the full attribute model is that the recall rate has increased from 0.375 to 0.563.

3.2.2 Belt Counter Testing

Belt Counter performed better than Palindrome in terms of percent correct. In the full attribute test, it had correctly classified 81.25% of the strands, while the selected attribute data set had 87.5%. Table 2 displays the confusion matrices between the two.

	Full Attributes		Selected Attributes	
Classified As ->	Aptamer	Non-Aptamer	Aptamer	Non-Aptamer
Aptamer	27	5	28	4
Non-Aptamer	7	25	4	28
Precision	0.794	0.833	0.875	0.875
Recall	0.844	0.781	0.875	0.875

Table 2. Confusion matrix comparing Belt Counter results with precision and recall divided by classification.

Once again, the confusion matrix shows the distribution of the classifications across both full attributes and selected attributes. It is interesting to note that the precision and recall do not have as big of a jump from the two models as Palindrome. In fact, the differences are rather minute. There was one more correctly identified aptamer, and three less misidentifications of non-aptamers.

3.2.3 Naïve Method Testing

Naïve Method did poorly when all the attributes were used. Once the selected attributes were used; however, the results greatly increased to be on par with the other two methods. Table 3 demonstrates the increase from full to only the selected attributes. Naïve Method classified aptamers correctly 56% of the time, but it did classify the actual aptamer strands 94% of the time.

	Full Attributes		Selected Attributes	
Classified As ->	Aptamer	Non-Aptamer	Aptamer	Non-Aptamer
Aptamer	30	2	30	2
Non-Aptamer	24	8	8	24
Precision	0.556	0.8	0.789	0.923
Recall	0.938	0.25	0.938	0.75

Table 3. Confusion matrix comparing Naïve Method results with precision and recall divided by classification.

The Naïve Method outperformed the Belt Counter in classifying the aptamer cases, and it outperformed Palindrome Counter in classifying the non-aptamer cases. It seems to have the mixed results of the two. With none of the methods doing clearly better compared to the other ones.

Chapter 4

ANALYSIS

In the end, Palindrome Counter, at its best, correctly identified 75% of the DNA strands. Belt Counter was able to correctly identify 87.5%. Naïve Method was able to correctly identify 84.4%. Their best performance all came from the Selected Attributes model. Table 3 shows the confusion matrices of the three methods for ease of comparison.

	Palindrome Counter		Belt Counter		Naïve Method	
Classified As ->	Aptamer	Non-Aptamer	Aptamer	Non-Aptamer	Aptamer	Non-Aptamer
Aptamer	30	2	28	4	30	2
Non-Aptamer	14	18	4	28	8	24
Precision	0.682	0.9	0.875	0.875	0.789	0.923
Recall	0.938	0.563	0.875	0.875	0.938	0.75

Table 4. A comparison of the best performing models for Palindrome, Belt and Naïve.

What can be easily gleaned from this table is that there is a difference in efficacy of the models by class. Specifically, while the Palindrome Counter model is able to correctly identify 30 aptamers, Belt Counter gets 28, and Naïve gets 30 as well. This seems appropriate enough though as it is only a 6% increase in the recall rate for

aptamers. The true difference comes when we look at non-aptamer classification. Palindrome Counter misclassified 14 non-aptamer strands while only doing that to two aptamer strands. Compared to Belt counter, where there were only four misclassifications for both, and the idea of the difference in models can be glimpsed. Belt Counter is able to better differentiate between random DNA strands and aptamers. Non-aptamer strands are better recognized and more likely to be correctly classified compared to Palindrome Counter. Belt Counter also beats out Naïve Method in the non-aptamer class but only by four cases. Belt Counter's effectiveness seems to be the same across both classes unlike Palindrome Counter.

The distinction between aptamers and non-aptamers for Palindrome Counter is not great enough. Conversely, Belt and Naïve both have pretty clear distinctions between the two classes. The aptamer classification in Belt Counter could use some improvement though to be as good as the Naïve Method. It is curious that Naïve Method usually did better than Palindrome and Belt Counter in the training experiment. This is accounted by the fact that there was no significant difference in percent correct between the data points. While the graphs show that Naïve Method did better, the significance testing shows that it probably was not the case. The training phase had non-aptamer strands that shared the same primers as the aptamer strands. This was not the case in the testing phase. In this light, one could argue that Palindrome Counter is able to detect the minute differences between different batches of DNA strands in SELEX sequencing. The same could be said for Naïve Method which solely relies on the singular position of particular bases. This is contrasted to Belt Counter which clearly is able to tell the difference between clearly disparate strands (random strands with no similar primers).

4.1 HYPOTHESIS RESULTS

The first hypothesis is weakly supported through the results of the second experiment. Using one of my counter methods, the model was able to better perform than chance or a naïve method. Unfortunately, the success is relegated to Belt Counter. Palindrome Counter only did as well as the Naïve Method in classifying aptamers. Belter Counter, while not doing as well in aptamers, classified the non-aptamers more accurately. And of course, both methods did better than chance in classifying.

As far as the second hypothesis, there is no evidence to suggest that using my counter methods will boost the performance of models across the board. The fact that there is no significant difference between Naïve Method and the two Counter methods show that performance does not necessarily increase across any method. While the two methods did perform better than chance in all methods, they failed to beat out the Naïve method at a significant level.

Thus, there is weak support for my thesis proposed. The results show that counting the bases does give some sort of reliable information for differentiating between aptamers and non-aptamers. The issue comes from the fact that most of the data had similar pieces of sequences as showed by ClustalW2. By having similar sequences, most of the strands would give similar count vectors and, in essence, not do much better than the Naïve method. The thesis, while grounded in previous research, might need a different implementation of it than the Palindrome or Belt Counter to show support for it.

Chapter 5

CONCLUSION

5.1 LIMITATIONS

The first limitation that can be easily seen is the fact that that both Counter programs require all of the aptamers to be of equal length. If they're not, then the data mining techniques used for them will not be applicable as many models do not handle missing values. The fundamental idea of the two Counter programs, that of counting base pairings, makes sense when all the strands of equal length: the particular position holds significance. It could be possible to do the same if the data were to be normalized along an average length of all the aptamers.

Going along with the length issue, the Counter programs are very sensitive to base shifts. For instance, given the possibility that one base is simply moved from one side to another, the count vector could change drastically for both sequences. Two aptamers differing by a few bases is clearly possible as the data I have indicates. Thus, there needs to be a way to account for simple base shifts not changing the whole result.

Another limitation is that there were no proper non-aptamers in the data set. Non-aptamers in this case were not actually tested against the specific target to see if they had any affinity for the target. In fact, the random strings I created for my non-aptamers could very well have had a few "aptamers" and there would be no way to know that. The

non-aptamers in the training data also suffered the same result. Unfortunately, the SELX method does not let researchers know which strands of DNA did not attach. Thus, the original data was formed by placing random sequences of bases in between the primer overhangs. Again, as mentioned with my data, there could be aptamers hidden within those sequences without knowing it. Given these issues, it is difficult to say whether one should bother to look at the non-aptamer classifier results. Until there is confirmed non-aptamers to test with, it will always be a cause for concern.

The final limitation is simply the size of the samples in question. Unfortunately, a lot of data had to be removed to stay with the equal length rule. More data would give more confidence to the ability of the models to classify based on the Counter methods.

5.2 FUTURE WORK

As mentioned with the limitation of size, there is a research group at The Ellington Lab at the University of Texas that is currently working on a database of all known aptamers [2]. At the time of this writing, the database is still not functioning, but it promises to deliver a wealth of aptamers that would be perfect for further research. It would be important to see if the two methods have different efficacy for the different types of aptamers. For example, it is possible that Palindrome Counter is better at approximating long, thin shaped RNA strands. Belt Counter could then conceivably be better at approximating strands that have more circular shapes. Certain molecules will accept different shapes more readily, and it is conceivable that the two methods might vary in efficacy depending on what type of molecule the aptamers are binding to.

It will also be important to get one more piece of information for the classifications. It will be a worthwhile effort to see how well the two methods work at

identifying aptamers if there was an index to indicate the affinity to bind to a certain molecule. That way, the models could be compared and set to only classify strands with a high enough index of affinity as aptamers. This could be used to fine-tune the methods to create models able to only choose very high-affinity bonding aptamers for certain molecules.

Finally, as mentioned in the hypothesis results, many of the DNA strands for the aptamers were very similar to each other. They had large chunks of the same sequences. To further test out my thesis, it would be important to see if there is a collection of aptamers with a very small amount of alignment. That way, the confounding variable can be taken out. In the given context, the Naïve Method would do the worst, consistently, because it would not be able to identify aptamers based on a singular base in a particular position. If there was access to such an aptamer collection of disparate sequences, then Palindrome and Belt Counter could both be put to a more rigorous test.

Appendix A

CLUSTALW2 ALIGNMENT RESULTS

```
MBE1A          -GGCATTACGGCCGGG-----TGT-C-GCTGATTT---CTGAATTTTG--------TGTG 41
MBE6A          -GGCATTACGGCCGGG-----TGT-CTGATTAATT---CTGATTATTG--------TGTG 42
MBE17A         -GGCATTACGGCCGGG-----TGT-TTGCTGATAT---CTGACTTTTC--------TCTG 42
MBE85          -GGCATTACGGCCGGG-----TGCGTTTTTGATAT---CTGA-TTGAC--------TTTG 42
MBE14A         -GGCATTACGGCCGGG-----TGCAGACT-GATTA---CTGACT-TTC--------TCTA 41
MBE36A         -GGCATTACGGCCGGG-----TGCAGACT-GATTA---CTGACT-TTC--------TCTA 41
MBE43A         -GGCATTACGGCCGGG-----TGCAGACT-GATTA---CTGACT-TTC--------TCTA 41
MBE81          -GGCATTACGGCCGGG-----CTTGTACT-GATTA---CTGACT-TTC--------TCTA 41
MBE33A         -GGCATTACGGCCGGG-----CGTATAAT-GATTA---CTGA---TTC--------TCTA 39
MBE62          -GGCATTACGGCCGGG-----TGCAGAATTGGAAA---CTG----TTA--------TCTA 39
MBE54          -GGCATTACGGCCGGG-----ATG-TTACTGACTTCTTCTGA---TT---------TCTA 41
MBE64          -GGCATTACGGCCGGG-----ATG-TTACTGACTTCTTCTGA---TT---------TCTA 41
MBE19A         -GGCATTACGGCCGGG-----AGACTTACTGATAT---CTGA---TTC--------TCTA 40
MBE52          GGGCATTACGGCCGGG-----ATT-TAACTGATTT---CTGAC--TTG--------TCTA 41
MBE45A         -ACTATAACGGCCGGGG----TGCAACTTTGATCT---CTTA-C-TTT--------TCTG 42
MBE55          -GGCATTACGGCCGGG-----TGTCGCACTGACTT---CTGATC-TTT--------TCTG 42
MBE11A         -GGCATTACGGCCGGGC----TGTTTACTGAACTA----TGAT-ACTT--------TCTG 42
MBE12A         -GGCATTACGGCCGGGC----TGTTTACTGAACTA----TGAT-ACTT--------TCTG 42
MBE39A         -GGCATTACGGCCGGGC----TGTTTACTGAACTA----TGAT-ACTT--------TCTG 42
MBE65          -GGCATTACGGCCGGGC----TGTTTACTGAACTA----TGAT-ACTT--------TCTG 42
MBE97          -GGCATTACGGCCGGGC----TGTTTACTGAACTA----TGAT-ACTT--------TCTG 42
MBE90          -GGCATTACGGCCGGG-----TGTTGAT--GACTA---CTGATTACTT--------TCTG 41
MBE72          -GGCATTACGGCCGGG-------TGTTTT-GACTG---ATGATTTTTGTA------TCTG 42
MBE106         -GGCATTACGGCCGGG-------TTTACT-GACTA---ATGATTTTTTAC------TCTG 42
MBE34A         -GGCATTACGGCCGGG-----TGCTGATT-GA-------TGATTTATGAC--AC--TAT- 41
MBE103         -GGCATTACGGCCGGG-----TGC---TT-GAC------TGACTTTTGAC-TAC--TTTT 41
MBE76          -GGCATTACGGCCGGGA----TGAT-T-T--A-------TGATTATTGACATAC--TCT- 41
MBE102         -GGCATTACGGCCGGG-----TGCT-TCT--AC------TGAAATCTGAC-TAC--TCT- 41
MBE98          -GGCATTACGGCCGGG-----TCTT-ACT-GAT------TGATTACTCAC--AC--TCT- 41
MBE79          -GGCATTACGGCCGGG-----TGTCGTACTGATTG---ATGATTTAC---------TCTG 42
MBE83          -GGCATTACGGCCGGG-----TGTCGTACTGATTG---ATGATTTAC---------TCTG 42
MBE2A          -GGCATTACGGCCGGG-----TGTCGTACTGATTG---ATGATTTAC---------TCTG 42
MBE42A         -GGCATTACGGCCGGG-----TGTCGTACTGATTG---ATGATTTAC---------TCTG 42
MBE51          -GGCATTACGGCCGGG-----TGTCGTACTGATTG---ATGATTTAC---------TCTG 42
MBE8A          -GGCATTACGGCCGGG-----TGTCGTACTGATTG---ATGATTTAC---------TCGG 42
MBE30A         -GGCATTACGGCCGGG-----TGCT-AACTGATTG---ATGAT-------------TTTG 37
MBE91          -GGCATTACGGCCGGG-----TGAC-TACTGATTG---ATGATATAT---------TCTG 41
MBE5A          -GGCATTACGGCCGGGA----TGGT-TACTGATTT---ATGATTTATC--------TCTG 43
MBE9A          -GGCATTACGGCCGGG------GCTACTT---------CTGATTGATTATT-TC--TCTG 41
MBE21A         -GGCATTACGGCCGGG------GCTACTT---------CTGATTGATTATT-TC--TCTG 41
MBE25A         -GGCATTACGGCCGGGT----AGCT-TTA---------CTGATTGATTAAT-TG---CTC 41
MBE47          -GGCATTACGGCCGGGT-----ACCACTTG--TTA---CTGATTACTGAA-------CTG 42
MBE96          -GGCATTACGGCCGGGT-----ACCATTT---------CTGATTACTTAAG-TG--CTTG 42
MBE48          -GGCATTACGGCCGGGT-----ACTACTGA--T-A---CTGATTTCTTGA-------CTG 41
MBE4A          -GGCATTACGGCCGGGGA---TGTT--------TA---ATGAACACTGA---TT--TCTA 40
MBE23A         -GGCATTACGGCCGGG-A---TG----------TA---CTGAAATCTGAAAATT--TCTA 40
```

```
MBE10A          -GGCATTACGGCCGGG-----TGCT-TAC----TA---CTGAA-ACTGA---TT--TTTC 40
MBE92           -GGCATTACGGCCGGG-----TGCTATAC----TA---CTGACTACTGA---TT----TC 40
MBE41A          -GGCATTACGGCCGGGA-----GTAACGCTA--TG---ATGATAAT-GTA------TCCC 42
MBE49           -GGCATTACGGCCGGGA-----GTAATGCTA--TG---ATGATAAT-GTG------TCCC 42
MBE104          -GGCATTACGGCCGGG-----TGCGATACTGAAT----ATGAAAGC----------TTTA 40
MBE112          -GGCATTACGGCCGGG-----TGCGATACTGAAT----ATGAAAGC----------TTTA 40
MBE44A          -GGCATTGCGGCCGGGGA---TGTAATATTGATCG---ATGAA-------------TTTG 40
MBE63           -GGCATTACGGCCGGG---------ATGTAGATCG---ATGAATACTGAG------TTTT 41
MBE7A           -GGCATTACGGCCGGGTG----CATGCTG--ATTA---CTGATT-TTAT-------CCTG 42
MBE20A          -GGCATTACGGCCGGGTG----CATGTTG--ATTA---CTGATT-T-AT-------CCTG 41
MBE13A          -GGCATTACGGCCGGGTG---CAATATAACTATGA---TTAAGTA-TAC-------TTTG 45
MBE100          -GGCATTACGGCCGGGTG---CAAACTGA--ATGA---TTAA-TA-TAC-------TCTG 42
MBE24A          -GGCATTACGGCCGGGT-----GCATCTA--ATGA---ATTCCTACTAT-------CCTG 42
MBE101          -GGCATTACGGCCGGGT-----AATCCAA--ATGA---CTAACTATTGA-------TCCG 42
MBE59           -GGCATTACGGCCGGG-----TGATGCAGAACTGA---TTGATTACT---------TCTG 42
MBE118          -GGCATTACGGCCGGG-----TGATGCAGAACTGA---TTGATTACT---------TCTG 42
MBE28A          -GGCATTACGGCCGGGCCA--TGATG-AAAACTGA---TT-ATGACT----------CTG 42
MBE38A          -GGCATTACGGCCGGG---------ATGTCA--TA---CTGATTACTGAT-GAC--TTT- 41
MBE50           -GGCATTACGGCCGGG---------ATGTCA--TA---CTGATTACTGAT-GAC--TTT- 41
MBE74           -GGCATTACGGCCGGG---------ATGTCA--TA---CTGATTACTGAT-GAA--TTTA 42
MBE77           -GGCATTACGGCCGGG---------ATGTT---TA---CTGATAACTGATTGAA--TCT- 41
MBE115          -GGCATTACGGCCGGG--------AATGCGG---A---CTGATTACTGAT--AC--TCT- 40
MBE117          -GGCATTACGGCCGGG--------AATGCGG---A---CTGATTACTGAT--AC--TCT- 40
MBE86           -GGCATTACGGCCGGG--------AATGCGG---A---CTGATTACTGAT--AC--TCT- 40
MBE40A          -GGCATTACGGCCGGG--------AATGCGG---A---CTGATTACTGAT--AC--TCT- 40
MBE46           -GGCATTACGGCCGGG--------AATGCGG---A---CTGATTACTGAT--AC--TCT- 40
MBE70           -GGCATTACGGCCGGGC-------AATGTCG-CTA---CTTAA-ACTGAT---C--TCC- 41
MBE110          -GGCATTACGGCCGGG--------AATGTT---TA---CTGAATACTGAT---C--AT-- 38
MBE107          -GGCATTACGGCCGGG--------AATGTCGATTA---CTGATTGCTGA----T--TCT- 41
MBE114          -GGCATTACGGCCGGG--------AATGTCGATTA---CTGATTGCTGA----T--TCT- 41
MBE113          -GGCATTACGGCCGGG--------AATGTCGATTA---CTGGTTGCTGA----C--TCT- 41
MBE26A          -GGCATTACGGCCGGG----------TGTCAAC------TGATTTATGA--CTG--TCT- 38
MBE68           -GGCATTACGGCCGGG----------TGTCTA-------TGATTAATGA--TTGGGTCT- 39
MBE31A          -GGCATTACGGCCGGG----------TGCTGG------CTGATTTCTGA-ATTG--TTT- 39
MBE32A          -GGCATTACGGCCGGG----------TGTCT-------CTGATTACTGATGGTG--TTT- 39
MBE84           -GGCATTACGGCCGGG----------TGCTGAACT--TAAGATTACTGA--TTC--T--- 40
MBE111          -GGCATTACGGCCGGG------------CTGA-----TATGATTACTGA--TTG--TTGA 38
MBE35A          -GGCATTACGGCCGGGG---------TGCCATT----ACTGATTGATGA---T---TGTT 40
MBE105          -GGCATTACGGCCGGG----------TGCCATT-----CTGATTG-------T---TGT- 33
MBE109          -GGCATTACGGCCGGG----------TGCCAA-----ACTGATTTCTGA---TA--TGTA 39
MBE15A          -GGCATTACGGCCGGG-----------GTTAAGCGG-TTTAATCATTGAG-----ATCTG 42
MBE87           -GGCATTACGGCCGGGAG----CCAGTATTA-------CTGATTGATGAT------TCTG 42
MBE108          -GGCATTACGGCCGGGAG----CCAGTATTA-------CTGATTGATGAT------TCTG 42
MBE3A           -GGCATTACGGCCGGGCGTCCCGTAGTGTCGGTAC----TGATTGATGT----------- 44
MBE22A          -GGCATTACGGCCGGGCGTCCCGTAGTGTCGGTAC----TGATTGATGT----------- 44
MBE27A          -GGCATTACGGCCGGGCGTCCCGTAGTGTCGGTAC----TGATTGATGT----------- 44
MBE37A          -GGCATTACGGCCGGGCGTCCCGTAGTGTCGGTAC----TGATTGATGT----------- 44
MBE53           -GGCATTACGGCCGGGCGTCCCGTAGTGTCGGTAC----TGATTGATGT----------- 44
MBE71           -GGCATTACGGCCGGGCGTCCCGTAGTGTCGGTAC----TGATTGATGT----------- 44
MBE82           -GGCATTACGGCCGGGCGTCCCGTAGTGTCGGTAC----TGATTGATGT----------- 44
MBE93           -GGCATTACGGCCGGGCGTCCCGTAGTGTCGGTAC----TGATTGATGT----------- 44
MBE29A          -GGCATTACGGCGGGGCGTCCCGTAGTGTCGGTAC----TGATTGATGT----------- 44
MBE57           -GGCATTACGGCCGGGCGTCCCGTAGTGTCGGTAC----TGATTAATGT----------- 44
MBE94           -GGCATTACGGCCGGGCGCCCCGTC-TGCCACCCTA---TCATCTCTGT----------- 44
                    **  **** ***

MBE1A           G--G--GGCGTCTTCTAG 55
MBE6A           G--G--GGCGTCTTCTAG 56
MBE17A          G--G--GGCGTCTTCTAG 56
MBE85           G--G--GGCGTCTTCTAG 56
MBE14A          G-TG--GGCGTCTTCTAG 56
```

```
MBE36A          G-TG--GGCGTCTTCTAG 56
MBE43A          G-TG--GGCGTCTTCTAG 56
MBE81           G-TG--GGCGTCTTCTAG 56
MBE33A          G-TGTGGGCGTCTTCTAG 56
MBE62           G-TG--GGCGTCTTCTAG 54
MBE54           G-TG--GGCGTCTTCTAG 56
MBE64           G-TG--GGCGTCTTCTAG 56
MBE19A          GGTG--GGCGTCTTCTAG 56
MBE52           G-TG--GGCGTCTTCTAG 56
MBE45A          G--G--GGCGTCTTCTAG 56
MBE55           G--G--GGCGTCTTCTAG 56
MBE11A          ---GGGG-CGTCTTCTAG 56
MBE12A          ---GGGG-CGTCTTCTAG 56
MBE39A          ---GGGG-CGTCTTCTAG 56
MBE65           ---GGGG-CGTCTTCTAG 56
MBE97           ---GGGG-CGTCTTCTAG 56
MBE90           ---GGGGGCGTCTTCTAG 56
MBE72           ---GGGG-CGTCTTCTAG 56
MBE106          ---TGGG-CGTCTTCTAG 56
MBE34A          ---GGGGGCGTCTTCTAG 56
MBE103          ---GGGGGCGTCTTCTAG 56
MBE76           ---GGGGGCGTCTTCTAG 56
MBE102          ---GGGGGCGTCTTCTAG 56
MBE98           ---GGGGGCGTCTTCTAG 56
MBE79           ---GG-GGCGTCTTCTAG 56
MBE83           ---GG-GGCGTCTTCTAG 56
MBE2A           ---GG-GGCGTCTTCTAG 56
MBE42A          ---GG-GGCGTCTTCTAG 56
MBE51           ---GG-GGCGTCTTCTAG 56
MBE8A           ---GG-GGCGTCTTCTAG 56
MBE30A          ---GG-GGCGTCTTCTAG 51
MBE91           ---GGTGGCGTCTTCTAG 56
MBE5A           ---GG-GGCGTCTTCTAG 57
MBE9A           ---GTGGGCGTCTTCTAG 56
MBE21A          ---GTGGGCGTCTTCTAG 56
MBE25A          ---GTGGGCGTCTTCTAG 56
MBE47           ---G-GGGCGTCTTCTAG 56
MBE96           ---G-GGGCGTCTTCTAG 56
MBE48           ---G-GGGCGTCTTCTAG 55
MBE4A           T--GGGGGCGTCTTCTAG 56
MBE23A          T--GGGGGCGTCTTCTAG 56
MBE10A          T--GGGGGCGTCTTCTAG 56
MBE92           T--AGGGGCGTCTTCTAG 56
MBE41A          G--G-GGGCGTCTTCTAG 57
MBE49           G--G-GGGCGTCTTCTAG 57
MBE104          G--GTGGGCGTCTTCTAG 56
MBE112          G--GTGGGCGTCTTCTAG 56
MBE44A          G--GTGGGCGTCTTCTAG 56
MBE63           G--G-GGGCGTCTTCTAG 56
MBE7A           G----GGGCGTCTTCTAG 56
MBE20A          G----GGGCGTCTTCTAG 55
MBE13A          G----GGGCGTCTTCTAG 59
MBE100          G----GGGCGTCTTCTAG 56
MBE24A          G----GGGCGTCTTCTAG 56
MBE101          G----GGGCGTCTTCTAG 56
MBE59           G----GGGCGTCTTCTAG 56
MBE118          G----GGGCGTCTTCTAG 56
MBE28A          G----GGGCGTCTTCTAG 56
MBE38A          ---GGGGGCGTCTTCTAG 56
MBE50           ---GGGGGCGTCTTCTAG 56
MBE74           ---GGGGGCGTCTTCTAG 57
```

```
MBE77           ---GGGGGCGTCTTCTAG 56
MBE115          ---GGGGGCGTCTTCTAG 55
MBE117          ---GGGGGCGTCTTCTAG 55
MBE86           ---GGGGGCGTCTTCTAG 55
MBE40A          ---GGGGGCGTCTTCTAG 55
MBE46           ---GGGGGCGTCTTCTAG 55
MBE70           ---GGGGGCGTCTTCTAG 56
MBE110          ---GGGGGCGTCTTCTAG 53
MBE107          ---GGGGGCGTCTTCTAG 56
MBE114          ---GGGGGCGTCTTCTAG 56
MBE113          ---GGGGGCGTCTTCTAG 56
MBE26A          CTGGTGGGGCGTCTTCTAG 56
MBE68           CTG-TGGGGCGTCTTCTAG 56
MBE31A          CTG-TGGGGCGTCTTCTAG 56
MBE32A          CTG-TGGGGCGTCTTCTAG 56
MBE84           CTG--GGGGCGTCTTCTAG 56
MBE111          CTG--GGGGCGTCTTCTAG 54
MBE35A          CTG--GGGGCGTCTTCTAG 56
MBE105          -----GGGGCGTCTTCTAG 46
MBE109          CTG-GGGGCGTCTTCTAG 56
MBE15A          G----GGGGCGTCTTCTAG 56
MBE87           T----GGGGCGTCTTCTAG 56
MBE108          T----GGGGCGTCTTCTAG 56
MBE3A           ------GGGCGTCTTCTAG 56
MBE22A          ------GGGCGTCTTCTAG 56
MBE27A          ------GGGCGTCTTCTAG 56
MBE37A          ------GGGCGTCTTCTAG 56
MBE53           ------GGGCGTCTTCTAG 56
MBE71           ------GGGCGTCTTCTAG 56
MBE82           ------GGGCGTCTTCTAG 56
MBE93           ------GGGCGTCTTCTAG 56
MBE29A          ------GGGCGTCTTCTAG 56
MBE57           ------GGGCGTCTTCTAG 56
MBE94           ------GGGCGTCTTCTAG 56
                      * **********
```

Appendix B

PALINDROME COUNTER PROGRAM

```
package palindromecount;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;
import java.util.Vector;

/**
 * This creates a vector of a given DNA strand and its classification.
 * @author Vladimir
 */
public class Main {

    public static void main(String[] args) throws IOException {
        Scanner input = new Scanner(System.in);
        System.out.println("What is the file location?");
        String fileName = input.nextLine();
        Scanner readFile = new Scanner(new File(fileName));
        FileWriter fw = new FileWriter("resultOfPali.csv");
        String seq = null, rev, classify = null;
        int i, j, k, count,diff, line = 0;

        while(readFile.hasNext()){
            line++;
            rev = readFile.nextLine();
            seq = rev.substring(0,rev.indexOf(","));
            classify = rev.substring(rev.indexOf(",")+1);
            System.out.println(seq+"\n"+classify);
            Vector<Integer> v = new Vector<Integer>();
```

```
            for(i=0, count = 0; i<seq.length()-1; i++){
                count = 0;
                j = i+1;
                k = i;
                while(k >= 0 && j < seq.length()){
                    diff = Math.abs(seq.charAt(k)-seq.charAt(j));
                    count += (diff==4 || diff==19)?1:0;
                    k--;
                    j++;
                }
                v.add(count);
            }
            for(i=0; i<v.size(); i++){
                fw.write(v.get(i)+",");
            }
            fw.write(classify+"\r");
        }
        fw.close();
        readFile.close();
    }
}
```

Appendix C

BELT COUNTER PROGRAM

```
package beltcount;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;
import java.util.Vector;

/**
 * This creates a vector version of a given DNA string and its class.
 * @author Vladimir
 */
public class Count {

    public static void main(String[] args) throws IOException {
        Scanner input = new Scanner(System.in);
        System.out.println("What is the file location?");
        String fileName = input.nextLine();
        Scanner readFile = new Scanner(new File(fileName));
        FileWriter fw = new FileWriter("resultOfBelt.csv");
        String seq = null, rev, classify = null;
        int i, j, k, count,diff, line = 0;

        while(readFile.hasNext()){
            line++;
            rev = readFile.nextLine();
            seq = rev.substring(0,rev.indexOf(","));
            classify = rev.substring(rev.indexOf(",")+1);
            System.out.println(seq+"\n"+classify);
```

```
            Vector<Integer> v = new Vector<Integer>();
            for(i=0, count = 0; i<seq.length()/2+1; i++){
               count = 0;
               j = i+1; //bottom start
               k = i; //top start
               //For palindrome count
               while(k >= 0 && j < seq.length()){
                  diff = Math.abs(seq.charAt(k)-seq.charAt(j));
                  count += (diff==4 || diff==19)?1:0;
                  k--;
                  j++;
               }
               //For other half
               k = seq.length()-1;
               while(k != j && j < seq.length()){
                  diff = Math.abs(seq.charAt(k)-seq.charAt(j));
                  count += (diff==4 || diff==19)?1:0;
                  k--;
                  j++;
               }
               v.add(count);
            }
            for(i=0; i<v.size(); i++){
               fw.write(v.get(i)+",");
            }
            fw.write(classify+"\r");
         }
         fw.close();
         readFile.close();
      }

}
```

Appendix D

DNA GENERATION PROGRAM

```
package dnacreator;

import java.io.FileWriter;
import java.io.IOException;
import java.util.Random;
import java.util.Scanner;

/**
 * This will create a list of randomly generated DNA sequences.
 * @author Vladimir
 */
public class generate {

    public static void main(String[] args) throws IOException {
        Scanner input = new Scanner(System.in);
        System.out.println("How many non-aptimers are needed?");
        int total = input.nextInt();
        System.out.println("How long should each be?");
        int length = input.nextInt();
        FileWriter fw = new FileWriter("dnaGenerate.csv");

        Random val = new Random(System.currentTimeMillis());
        int nextVal = 0;
        char[] letters = {'A','T','C','G'};

        for(;total > 0; total--){
            for(int i = 0; i<length;i++){
                nextVal = val.nextInt(4);
                fw.write(letters[nextVal]);
            }
```

```
            fw.write(",NON-APTIMER\r");
        }

        fw.close();
    }

}
```

Appendix E

TRAINING FULL RESULTS

```
Tester:     weka.experiment.PairedCorrectedTTester
Analysing:  Percent_correct
Datasets:   7
Resultsets: 3
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       5/7/12 8:11 AM


Dataset                   (1) naivemet | (2) train (3)
train
----------------------------------------------------------
-
bayes.NaiveBayes '' 5.995(100)   95.58 |   94.43     94.20
functions.SimpleLogistic (100)   92.54 |   90.97     89.66
functions.MultilayerPerce(100)   92.70 |   89.61     91.68
meta.Dagging '-F 10 -S 1 (100)   88.10 |   88.59     89.42
rules.DecisionTable '-X 1(100)   85.53 |   76.13     80.44
trees.J48 '-C 0.25 -M 2' (100)   93.18 |   87.86     88.62
trees.RandomForest '-I 10(100)   96.23 |   90.55     89.80
----------------------------------------------------------
-
                                 (v/ /*) |   (0/7/0)
(0/7/0)

Tester:     weka.experiment.PairedCorrectedTTester
Analysing:  Percent_correct
Datasets:   7
Resultsets: 3
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       5/7/12 8:11 AM


>-<   >   < Resultset
```

```
  0   0   0 trainingPali
  0   0   0 trainingDataBelt
  0   0   0 naivemethodTrain

Key:
(1) naivemethodTrain
(2) trainingDataBelt
(3) trainingPali

Tester:     weka.experiment.PairedCorrectedTTester
Analysing:  Percent_correct
Datasets:   3
Resultsets: 7
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       5/7/12 8:24 AM


Dataset                   (1) bayes.Na | (2) funct (3)
funct (4) meta. (5) rules (6) trees (7) trees
----------------------------------------------------------
----------------------------------------
naivemethodTrain         (100)   95.58 |   92.54     92.70
88.10 *   85.53 *   93.18     96.23
trainingDataBelt         (100)   94.43 |   90.97     89.61
88.59     76.13 *   87.86     90.55
trainingPali             (100)   94.20 |   89.66     91.68
89.42     80.44 *   88.62     89.80
----------------------------------------------------------
----------------------------------------
                                (v/ /*) |   (0/3/0)
(0/3/0)   (0/2/1)   (0/0/3)   (0/3/0)   (0/3/0)


Key:
(1) bayes.NaiveBayes '' 5.9952312017856973E18
(2) functions.SimpleLogistic '-I 0 -M 500 -H 50 -W 0.0'
7.3977106263047055E18
(3) functions.MultilayerPerceptron '-L 0.3 -M 0.2 -N 500 -V
0 -S 0 -E 20 -H a' -5.9906078170482104E18
(4) meta.Dagging '-F 10 -S 1 -W functions.SMO -- -C 1.0 -L
0.0010 -P 1.0E-12 -N 0 -V -1 -W 1 -K
\"functions.supportVector.PolyKernel -C 250007 -E 1.0\"'
4.5601658765700741E18
(5) rules.DecisionTable '-X 1 -S \"BestFirst -D 1 -N 5\"'
2.8885570781657011E18
```

```
(6) trees.J48 '-C 0.25 -M 2' -2.17733168393644448E17
(7) trees.RandomForest '-I 10 -K 0 -S 1'
4.21683947075142 86E18

Tester:     weka.experiment.PairedCorrectedTTester
Analysing:  Percent_correct
Datasets:   3
Resultsets: 7
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       5/7/12 8:25 AM


>-<   >   < Resultset
  4   4   0 bayes.NaiveBayes '' 5.9952312017856973E18
  3   3   0 trees.RandomForest '-I 10 -K 0 -S 1'
4.21683947075142 86E18
  2   2   0 trees.J48 '-C 0.25 -M 2' -
2.17733168393644448E17
  2   2   0 functions.MultilayerPerceptron '-L 0.3 -M 0.2 -
N 500 -V 0 -S 0 -E 20 -H a' -5.9906078170482104E18
  2   2   0 functions.SimpleLogistic '-I 0 -M 500 -H 50 -W
0.0' 7.3977106263047055E18
  0   2   2 meta.Dagging '-F 10 -S 1 -W functions.SMO -- -C
1.0 -L 0.0010 -P 1.0E-12 -N 0 -V -1 -W 1 -K
\"functions.supportVector.PolyKernel -C 250007 -E 1.0\"'
4.5601658765700741E18
-13   0  13 rules.DecisionTable '-X 1 -S \"BestFirst -D 1 -
N 5\"' 2.8885570781657011E18
```

Appendix F

TRAINING LESS RESULTS

```
Tester:     weka.experiment.PairedCorrectedTTester
Analysing:  Percent_correct
Datasets:   7
Resultsets: 3
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       5/7/12 8:45 AM


Dataset                   (1) 'naiveme | (2) 'trai (3)
'trai
-------------------------------------------------------------
-
bayes.NaiveBayes '' 5.995(100)   95.37 |   95.49     97.39
functions.SimpleLogistic (100)   93.13 |   91.85     91.95
functions.MultilayerPerce(100)   93.48 |   90.96     92.91
meta.Dagging '-F 10 -S 1 (100)   88.54 |   91.42     90.88
rules.DecisionTable '-X 1(100)   87.02 |   75.81 *   79.05
trees.J48 '-C 0.25 -M 2' (100)   94.10 |   88.23     88.98
trees.RandomForest '-I 10(100)   96.16 |   90.35     92.02
-------------------------------------------------------------
-
                                 (v/ /*) |   (0/6/1)
(0/7/0)


Key:
(1) 'naivemethodTrain-unsupervised.attribute.Remove-R1-6,8-
31,33-34,36-37,40-41,46-56'
(2) 'trainingDataBelt-unsupervised.attribute.Remove-R1-
5,9,11-13,19,22-23,25-29'
(3) 'trainingPali-unsupervised.attribute.Remove-R1-5,7-
14,16,18,20-23,25-33,37,39,41-42,45-46,51-55'

Tester:     weka.experiment.PairedCorrectedTTester
Analysing:  Percent_correct
Datasets:   7
```

```
Resultsets: 3
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       5/7/12 8:47 AM


>-<   >   < Resultset
  1   1   0 'naivemethodTrain-
unsupervised.attribute.Remove-R1-6,8-31,33-34,36-37,40-
41,46-56'
  0   0   0 'trainingPali-unsupervised.attribute.Remove-R1-
5,7-14,16,18,20-23,25-33,37,39,41-42,45-46,51-55'
 -1   0   1 'trainingDataBelt-
unsupervised.attribute.Remove-R1-5,9,11-13,19,22-23,25-29'

Tester:     weka.experiment.PairedCorrectedTTester
Analysing:  Percent_correct
Datasets:   3
Resultsets: 7
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       5/7/12 8:48 AM


Dataset                   (1) bayes.Na | (2) funct (3)
funct (4) meta. (5) rules (6) trees (7) trees
----------------------------------------------------------
----------------------------------------
'naivemethodTrain-weka.fi(100)   95.37 |   93.13     93.48
88.54 *   87.02 *   94.10     96.16
'trainingDataBelt-weka.fi(100)   95.49 |   91.85     90.96
91.42     75.81 *   88.23     90.35
'trainingPali-weka.filter(100)   97.39 |   91.95 *   92.91
90.88 *   79.05 *   88.98 *   92.02
----------------------------------------------------------
----------------------------------------
                                (v/ /*) |   (0/2/1)
(0/3/0)   (0/1/2)   (0/0/3)   (0/2/1)   (0/3/0)


Key:
(1) bayes.NaiveBayes '' 5.9952312017856973E18
(2) functions.SimpleLogistic '-I 0 -M 500 -H 50 -W 0.0'
7.3977106263047055E18
(3) functions.MultilayerPerceptron '-L 0.3 -M 0.2 -N 500 -V
0 -S 0 -E 20 -H a' -5.9906078170482104E18
```

```
(4) meta.Dagging '-F 10 -S 1 -W functions.SMO -- -C 1.0 -L
0.0010 -P 1.0E-12 -N 0 -V -1 -W 1 -K
\"functions.supportVector.PolyKernel -C 250007 -E 1.0\"'
4.5601658765700741E18
(5) rules.DecisionTable '-X 1 -S \"BestFirst -D 1 -N 5\"'
2.8885570781657011E18
(6) trees.J48 '-C 0.25 -M 2' -2.17733168393644448E17
(7) trees.RandomForest '-I 10 -K 0 -S 1'
4.21683947075142 86E18

Tester:     weka.experiment.PairedCorrectedTTester
Analysing:  Percent_correct
Datasets:   3
Resultsets: 7
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       5/7/12 8:48 AM


>-<   >   < Resultset
  7   7   0 bayes.NaiveBayes '' 5.9952312017856973E18
  4   4   0 trees.RandomForest '-I 10 -K 0 -S 1'
4.21683947075142 86E18
  2   2   0 functions.MultilayerPerceptron '-L 0.3 -M 0.2 -
N 500 -V 0 -S 0 -E 20 -H a' -5.9906078170482104E18
  1   2   1 trees.J48 '-C 0.25 -M 2' -
2.17733168393644448E17
  1   2   1 functions.SimpleLogistic '-I 0 -M 500 -H 50 -W
0.0' 7.3977106263047055E18
 -1   2   3 meta.Dagging '-F 10 -S 1 -W functions.SMO -- -C
1.0 -L 0.0010 -P 1.0E-12 -N 0 -V -1 -W 1 -K
\"functions.supportVector.PolyKernel -C 250007 -E 1.0\"'
4.5601658765700741E18
-14   0  14 rules.DecisionTable '-X 1 -S \"BestFirst -D 1 -
N 5\"' 2.8885570781657011E18
```

References

[1] *Reviews Glossary*. (n.d.). Retrieved October 25, 2010, from Nature: http://www.nature.com/nrc/journal/v4/n10/glossary/ nrc1456_glossary.html

[2] About. (2006). Retrieved October 25, 2010, from The Ellington Lab: http://aptamer.icmb.utexas.edu/index.php

[3] Raghavendra Joshi, H. J.-A. (2008, November 18). Selection, characterization, and application of DNA aptamers for the capture. Retrieved October 25, 2010, from Molecular and Cellular Probes: http://static.msi.umn.edu/rreports/2009/7.pdf

[4] Zuker, M. Stiegler, P. (1981, November 1). Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. Retrieved May 4, 2012, from NCBI: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC326673/pdf/nar00394-0137.pdf

[5] Systematic Evolution of Ligands by Exponential Enrichment. (2010, September 30). Retrieved October 25, 2010, from Wikipedia: http://en.wikipedia.org/wiki/Systematic_evolution_of_ligands_by_exponential_enrichment

[6] Polymerase Chain Reaciton. (2012, May 4). Retrieved May 4, 2012, from Wikipedia: http://en.wikipedia.org/wiki/Polymerase_chain_reaction

[7] Sampson, T. (2003, April 3). Aptamers and SELEX: the technology. Retrieved May 5, 2012 from ScienceDirect: http://www.sciencedirect.com/science/article/pii/S0172219003000358

[8] Christopher G. Knight, M. P. (2008, October 23). Array-based evolution of DNA aptamers allows modeling of an explicit sequence-fitness landscape. Retrieved October 25, 2010, from Nucleic Acids Research: http://nar.oxfordjournals.org/content/37/1/e6.full

[9] MicroRNA. (2012, May 6). Retrieved May 6, 2012, from Wikipedia: http://en.wikipedia.org/wiki/MicroRNA

[10] Graham, J. Zarbl, H. Use of Cell-SELEX to Generate DNA Aptamers as Molecular Probes of HPV-Associated Cervical Cancer Cells. Retrieved May 6, 2012 from PLoS One: http://www.plosone.org/article/info%3Adoi%2F10.1371%2Fjournal.pone.0036103

[11] Vinkenborg, J. Karnowski, N. Famulok, M. (2011, July 18). Aptamers for allosteric regulation. Retried May 6, 2012 from Nature: http://www.nature.com/nchembio/journal/v7/n8/full/nchembio.609.html

[12] Ji ang, P. Haonan, W. Wang, W. Ma, W. Sun, X. Lu, Z. (2007, April 26). MiPred: classification of real and pseudo microRNA precursors using random forest prediction model with combined features. Retrieved May 5, 2012 from Oxford Journals: http://nar.oxfordjournals.org/content/35/suppl_2/W339.full

[13] Ding, J. Zhou, S. Guan, J. (2011, May 28). miRFam: an effective automatic miRNA classification method based on n-grams and a multiclass SVM. Retrieved May 5, 2012 from PubMed: http://www.ncbi.nlm.nih.gov/pubmed/21619662

[14] Konig, J. Julius, C. Buamann, S. Homann, M. Goringer, U. Feldbrugge, M. (2007, February 5). Retrieved May 5, 2012 from Facultad De Ciencias: http://lim.fcien.edu.uy/pag_anterior/regulacion_2008/articulos/Koning%20RNA%202007.pdf

[15] Zheng, J. Hao, J. Li, Z. Yan, Q. Wang, J. Han, F. Li, Y. Lu, Z. Su, Y. (2011, October 2011). Comparison of classifications of aptamers against Vibrio alginolyticus based on their primary and secondary structure. Retrieved May 5, 2012 from Academic Journals: http://www.academicjournals.org/ajb/PDF/pdf2012/5Jan/Zheng%20et%20al%201.pdf

[16] Condon, A. Problems on RNA Secondary Structure Prediction and Design. Retrieved May 6, 2012 from University of British Columbia: http://www.cs.ubc.ca/~condon/papers/icalp03.pdf

[17] Dunham, Margaret, H. (2003). *Data Mining Introductory and Advanced Topics*. Upper Saddle River, NJ: Prentice Hall.

[18] ClustalW2 – Multiple Sequence Alignment. Retrieved May 6, 2012, from EMBL-EBI: http://www.ebi.ac.uk/Tools/msa/clustalw2/

[19] Online Calculator: Levenshtein Distance. Retrieved May 6, 2012, from PlanetCalc: http://planetcalc.com/1721/

[20] Jovanovic, V. (2010, October 26). Classification of Aptamers: Project 1.

[21] Weka. Retrieved May 4, 2012 from The University of Waikato: http://www.cs.waikato.ac.nz/ml/weka/

[22] Jovanovic, V. Dunham, M. (2011, February 14). Unique base pair count vectors in analysis of nucleotide sequences, *poster*.

[23] Dagging. Retrieved May 6, 2012, from Sourceforge: http://weka.sourceforge.net/doc.packages/dagging/weka/classifiers/meta/Dagging.html