5-5-2017

Formal Modeling and Mining of Big Data

Aljoharah A. Algwaiz
University of Connecticut - Storrs, aljoharah.algwaiz@uconn.edu

Formal Modeling and Mining of Big Data

Aljoharah A. Algwaiz , Ph.D.

University of Connecticut, 2017

Abstract

As data collection technologies are advancing and memory storage costs are declining, volumes of data collected have soared. Scientists and investigators are collecting all possible data in fear of missing out on important information. With the merge of the data collection trend, researchers were studying data mining and analysis to find the most efficient way to data mine. There are various valuable data mining techniques that can be found in literature such as Support Vector Machine (SVM), Neural Networks (ANN), and Formal Methods (Grammars) [7]. Grammars are a very valuable in analyzing structured data and describing them in a condense matter. However, not many have used it for data mining even though it has many benefits [1]. In this research we present an approach to data mine big data. First, a grammar is inferred to build a structural model that describes the data. Then, on the next phase, a probabilistic context-free grammar is inferred and a model for a more complex structures. Given an input sequence, the model parses and generates the probability of that data sequence being part of the class based on its structural characteristics.

Grammatical concatenation is utilized in case of existing sub-structures within the class's structural description. The model then accepts, or rejects, the input as part of the data's class by comparing the probability to a pre-set threshold. Finally, this is applied on a heterogeneous large data set by inferring multiple grammars. After building grammatical model for each class, the algorithm parse multiple points in the large set. It then classifies these data into smaller sets where they share similar structural characteristics using probabilistic grammar. If more than one class accepts the data point, it is associated to the highest ranking class. Biological data, DNAs and Proteins, were used for experimentation in this research.

Formal Modeling and Mining of Big Data

Aljoharah A. Algwaiz

M.S., University of Connecticut, 2016

M.B.A, King Saud University, 2009

B.S., King Saud University, 2005

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

at the

University of Connecticut

2017

APPROVAL PAGE

Doctor of Philosophy Dissertation

Formal Modeling and Mining of Big Data

Presented by
Aljoharah Algwaiz, M.S., M.B.A., B.S.

Co-Major Advisor _____
Reda Ammar

Co-Major Advisor _____
Sanguthevar Rajasekaran

Associate Advisor _____
Swapna Gokhale

Associate Advisor_____
Yufeng Wu

University of Connecticut

2017

Acknowledgment

First of all, I would like to thank Allah for providing me with health, patience, and knowledge to make this possible.

I was highly privileged to have both Prof. Reda Ammar and Prof. Sanguthevar Rajasekaran as my major advisors. I would like to express my appreciation and gratitude for your continuous guidance, mentoring, and encouragement. I learned a lot by working closely with you as a student, both academically and professionally.

I would like to express my gratitude to my family. Without you, this wouldn't be possible; My loving mother (Huda), my supportive father (Abdullah), my beloved husband (Riyad), my siblings (Mona, Mohammed, Nora, and Abdulrahman), and my dear children (Rashed and Nora). Thank you for your patience, prayers, encouragement, and unconditional love. You are my sunshine on a rainy day.

A special thanks and appreciation to my mentor and dear Uncle, Dr. Tawfig Alrabiah. Thank you for the valuable career and personal support and advice throughout these years.

I would also like to thank all my friends for supporting and motivating me throughout this journey to thrive and accomplish my goals.

Finally, I would like to thank the Saudi Arabian Cultural Mission (SACM) and King Saud University (KSU) for the full scholarship opportunity.

Contents

Chapter 1

Introduction

Nowadays, data is available everywhere and data collection became easier and cheaper. This simulated a need to collect and store all these data in fear of missing out on important information. Various data storage solutions emerged such as cloud computing, distributed databases, data warehouses, and big data. This caused an explosion in data collection. The decrease in data storage and computer power costs made it possible to do that. A report published by IDC [30] showed that the data will grow from 130 exabytes (EB) to 40,000 EB (40 trillion GB) from 2005 to 2020. This will amount to more than 5,200 GB per person. In order to be able to use and utilized these large volumes of valuable data, an efficient and feasible data mining algorithm is needed. There is a need for a data mining tool that can automatically extract, classify, and identify significant and relevant information. Data mining is vital for decision making, understand underlying patterns, and forecast future trends. The term “Big Data” was believed to be first introduced in 1998 by John Mashey. Then, the term wbecame wide spread as of 2011 and technologies rapidly advanced in this field [29]. Data mining is used in a wide variety of fields such as business intelligence, biotechnology, multimedia, scientific discoveries, and internet navigation [45]. Therefore, we see investments in data mining growing fast [30].

When data mining, one must consider the approach’s efficiency, accuracy, and security. Efficiency here means that it uses minimal space and time. However, it also

has to be accurate, as it must generate the expected results. With large volumes of data comes also the challenge of keeping data that might be sensitive secure and not expose a security threat [28]. This made data mining big data a very attractive topic for researchers and scientists.

1.1. Problem Definition and Motivation

The simple definition of data mining is an automated technique used to understand and extract useful information from raw data [45]. Data mining has been around for many years. With the introduction of big data, it is even more prevalent. Traditionally, data mining used to be done manually through statistics and econometrics to extract findings within big data. As more data is collected and stored, these approaches proved to be unreliable, inaccurate and time-consuming. It also confronts the challenge of scalability, security, and efficiency [1]. As data storages grew bigger and became more complex and diverse, manually data mining was infeasible. Today, the data mining process is automated through applying machine learning algorithms to data mine data. Several valuable data mining algorithms are available in literature such as; K-Nearest Neighbor (KNN), Neural Networks, Bayesian Networks (BNs), Support Vector Machine (SVM), Decision Trees, Formal Methods (Grammars), and others [7]. However, few use formal methods to data mine. These techniques provided a faster, more efficient and more reliable way for data mining. In this paper, we focus on data mining using formal methods (Grammars).

Although not many researchers focus on formal methods, there are several benefits of using formal grammars for data mining [1]. Formal grammar is an effective and

advanced tool for data association, extraction, and modeling. Formal methods have various qualities that make them an attractive research topic. Formal grammars can deliver a statistical and structural description of the data in a condensed matter. It is also capable of applying highly-integrated data mining with capabilities from data processing to a macro data analysis using a common programming language. When structured data are presented as sequences, formal grammars can overcome location specific structural characteristics. Using formal grammars can also assist in predicting and associating additional data that belongs to the same class. This makes it suitable to be used to a wide-range of structured data classes.

This dissertation is conducted with two main objectives in mind;

1- Utilizing formal methods to develop formal data models. This objective focuses on building a model for a family class that exists within the big data. The model embodies the structural and statistical characteristics of the family class. The models also functions as an input filter. It accepts and rejects given input as part of the family class based on its structural and statistical features. Various formal data models are developed for big data containing various classes.

2- Constructing a model based data mining algorithm. After developing a formal data model for each family class in the big data, the aim is developing a data mining algorithm to exploit the developed models. The model-based data mining algorithm classifies the big data into smaller sets. Data within each set share similar structural and statistical characteristics that signify the class. If

new data are introduced to the big data, the algorithm associates it to the appropriate class set.

The process of accomplishing these objectives, the work is divided into four topics;

A) Learning a structural model

Learning a structural model that can represent a data class. A model provides data description in a compact form. Structural models here are built using formal methods through grammatical inference. When inferring and choosing a suitable grammar for the model two challenges are addressed; over-fitting and over-generalization. A grammar is said to be over-fitting when it represents exactly one input sequence. An example of such grammatical inference tool that generates an over-fitting grammar is Sequitur [54]. To the contrary, a grammar can be over-generalized. A grammar over-generalization means the grammar is very broad that it accepts inputs that are not part of the true grammar. This problem can lead to a high percentage of false positives.

B) Data association

Associating a data sequence to a family class by grammatically parsing the sequence using the structural model. The model identifies the structural features that signify the class's structure. It then generates a probability that a sequence is part of the grammar. If the probability is less than a certain threshold, the sequence is rejected, otherwise accepted.

Data association confronts the challenge of a data class containing a variety of structures with different lengths. Identifying significant a structural feature that is

not fixed in location can lead to misidentification. An example of this challenge is distinguishing a real TATA-box in a DNA sequence from a gap that contains a high TATA sequence repetition without depending on location.

C) Data classification

Given a heterogeneous big data, the aim is to classify the big data into smaller sets where each set represent a family class. For each class a grammar is inferred then a grammatical model is developed. The big data is injected into an algorithm that employs these models. One set represents all sequences that were accepted by a certain structural model. This step faces the challenges of classifying a sequence to the correct class when it is accepted by more than one structural model. Also, when classifying big data into smaller clusters, it is necessary to balance partitioning. This is important so no one grammatical model accepts very limited number of sequences while another grammatical model accepts all sequences in the database.

D) Building a composite model for big data complex sequences

The aim is to build a grammatical model that is able to data mine not only big data, but complex data structures as well. By complex structures we mean sequences that their identification requires more analysis and depth. For instance, identification of some significant patterns as significant might not be as simple as studying the patterns. It may require studying the patterns before and

after that pattern. Also, it may depend of the existence, or absence, of another pattern. Example of such sequence is DNA and RNA sequences.

Each of these topics is going to be further elaborated on in more detail in separate chapters later in this dissertation.

In this investigation, the main data domain is intended for structural data. Therefore, biomedical data such as DNA and protein sequences where chosen for experimentation. Based on Noam Chomsky different formal grammar classifications [17], formal methods in this research were used up to Context-Free Grammar. The approach is not limited to a single programming language. It can be implemented in most common programming languages such as C++ and Java. Complex and large data structures were handled through segmentation and concatenation [5].

1.2. Dissertation Organization

This dissertation has seven chapters. This chapter provided an introduction for dissertation research's, motivation, and problem definition. The next chapter explains fundamental definitions related to this research. Next, the four main topics will be investigated. The third chapter covers the topic of learning a structural model. Data association is presented in chapter four. In chapter five, the third topic is introduced, Data classification, which involves classifying big data using formal methods. Then, the building a composite model for big data complex sequences is explained in chapter six. Finally, chapter seven concludes this dissertation, which presents this dissertation's summary, contributions, and future work.

Chapter 2

Formal Grammars and Data Mining

2.1. Introduction

Grammatical inference is a very useful tool to model a family of structured sequences. It can be used for pattern recognition, user behavioral study and prediction, classification, and translation. In linguistics, probabilistic context free grammars (PCFG) are used for identifying sentence structures for applications such as translation and error detection [33]. Grammars are also used in bioinformatics to analyze and identify various kinds of DNA and RNA structures [53]. It can assist in identifying certain significant regions and sequence type. In behavioral analysis, grammars can be useful in decision making and analysis. A certain business can analyze different customer buying habits and target marketing and restocking orders based on those analysis [25]. A webmaster can analyze visitors navigation patterns and arrange the site's design and structure for a better user experience [12].

These are just some examples of applications where formal grammars can be utilized in different fields. However, they is not limited to those areas. Formal grammars are very attractive as they can be applied to describe a wide range of structural data types.

2.2. Formal Grammars

2.2.1.Definition

Grammars are defined as syntax rules used to describe the data's structural relations of patterns or the syntax of languages [26]. A grammar G defines the language L(G).

L(G) consists of finite or infinite set of sentences or sequences belonging to the same language or class. All sentence from the same language consists of structural features that characterizes the language [26].

In general, formal grammars consists of four tuples G=<T, N, R, S> where:

T set is terminal symbols,

Nset of non-terminal symbols,

S start symbol (S ∈ N), and

R set of rules that govern the grammar (also called production rules).

The grammatical rules in R have the format A → B C such that A ∈ N and a, b ∈ T or N.

To provide a better understanding, an example of a grammar from a paper by Fu, King-Sun, and Booth [26] that shows the way grammatical rules and symbols are expressed in grammars. The example shows a grammar for a sequence of a's followed by a single b then an equal number of a's. The grammatical rules will be as follows:

S → a S a

S → b

It will generate an infinite loop that generates sequences belonging to the grammar such as {aba, aabaa, aaabaaa, ...}. The grammatical language can be expressed as L(G)= { $a^n ba^n$ }. the sequences such as {a, aab, ba} will be rejected by the grammar.

2.2.2 Operations

To describe more complex structures, some operations are used grammars. We have

a grammar G1 and G2 which are used to represent a language L1 and L2 respectively "G1= G(L1) and G2 = G(L2) ". Operations are applied on these grammars as follows:

1) Union: When having several languages {L1, L2, .., Ln}, the union operation is applied on these languages to combines them in a single language L [5]. The "∪ " shape is the symbol for the union operation.

   L = L1 ∪ L2 ∪ …. ∪ Ln

   The total strings S in the language L contains various sub-sets of strings S1, S2, .. Sn where Si ∈ Li.

   The union of two languages L1 and L2 is the union of their grammars will result a new language L such that :

   L1 ∪ L2 = G(L1)∪ G(L2)

   The start rule for the new language after the union operation will be

   S→ S1

   S→S2

2) Concatenation: The concatenation operation is similar to the union operation. It is different where the concatenation of L1, L2, .. , Ln forms a new language L [5]. The concatenation operation does not have a symbol. A new string S can be created by simply concatenating several sub-strings together.

   S= S1 S3 S2 S1 S8 where Si ∈ Li.

   The concatenation of two languages L1 and L2 is equal to the concatenation of their grammars and will produce a new language L such that:

L= L1 L2 = G(L1) G(L2)

The start rule for the new language after concatenation will be

S→ S1S2

2.2.3 Probabilistic Context-Free Grammar

Probabilistic grammar is similar to the previous general grammar definition discussed in previous section. However, probabilistic grammar has five tuples with an additional set P. G ={T, N, S, R, P}. P is the set of rewrite rule probabilities with 1-1 association to the rewrite rules in R.

To turn a grammar G into a probabilistic grammar, the probability of each rule R_A for each non-terminal A ∈ N is calculated so that:

- The total probabilities of rewrite rules, with non-terminal A in the right-hand side, will exactly be equal to 1. $\Sigma\, P(A) = 1$.
- The value of each one of these probabilities in each rule is not be less than 0 and not greater than 1 (0 <= P(A) <= 1).

2.2.4. Applications

There are various features that make formal grammars an attractive tool. Formal grammars are able to provide a structural and statistical description of the data in a condense matter [1]. It can also be used as a syntactic source to generate all patterns belonging to a specific class (finite and infinite) [26]. In addition, they can predict additional phenomena and associate it to a data class [26].

Formal methods are successfully being used in a verity of fields such as natural

language processing [42], bioinformatics [53], and applied behavior analysis [6]. They proved to be effectual in describing the syntax of a language or the structural relations in patterns or data [26].

Various researchers were interested in grammars in several fields. Since simple grammars cannot solve complex structures, various classes of grammars were found to solve different problems. In 1956, Chomsky introduced a formal grammar hierarchy that consists of four levels [17]. The simplest form of grammar is the finite state grammar. Most researchers focused their attention on finite state grammar [5]. Another extension of this grammar is the context-free grammar. During research, it has found that the sample used has statistical characteristics. Therefore, this research will cover formal grammars up to probabilistic context-free grammar.

2.2.5. Grammatical Inference Methodologies

The process of learning a grammar is called grammatical inference (also called grammatical induction). It is defined as learning the set of syntactic rules that governs the structural characteristics of sentences or patterns from a finite set of sample sentences [22]. Learning can be done from a finite sample of positive sentences S^+ belonging to the grammar's language L(G) [26]. Learning can also be done using an additional negative finite sample S^- to exclude sentences that do not belong to the language L(G). negative sample a S^- are used to better refine the grammar and reduce over-generalization.

The grammatical inference process consists of three main steps in general [5]:

1- Identifying and analyzing targeted data using a sample training set. The alphabets that make up a sentence are the terminal symbols. The relationships between these symbols are the production rules. A suitable grammar type is identified based on the structural nature of the sentences in the targeted language.

2- Choose an appropriate grammar level and infer a grammar using a sample training set.

3- Testing the goodness of the generated grammar using a different testing set. Validating the results using the appropriate fitness measurement. Process may be repeated until testing results are satisfactory. Then, the inferred grammar can be deployed (Figure 2.1).

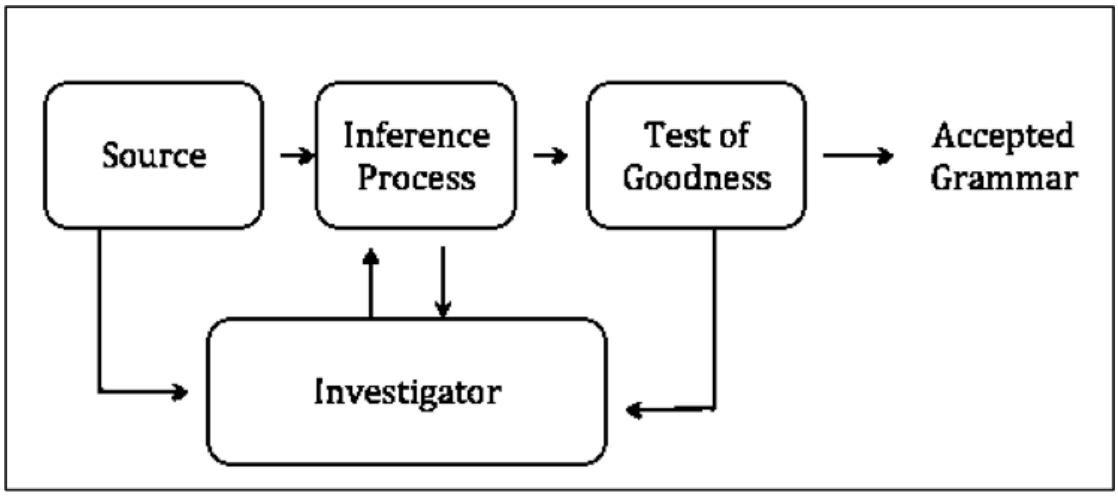


Figure 2.1: Grammatical Inference Process [5]

Grammatical inference has been used in various fields such a speech recognition, linguistics, bioinformatics, and behavioral analysis. However, it is until recently that it has attracted researchers in the data mining field.

2.3 Data Mining

Data mining is the process of studying and analyzing data from different perspectives with the aim of learning new useful information. Data mining is vital tool used in various different fields. It

To data mine, there are several solutions proposed. Some of the most popular ones are [7]:

-**Support vector machine (SVM)** takes labeled training examples and tries to generate new data that belongs one class or the other. It represents the data as points in space and tries to maximize the gap between the two classes [41].

-**Artificial Neural Networks (ANN)** is inspired by the human brain. It self-learns from examples. It is a network consists of highly interconnected processing elements that work in parallel. Then, it can identify new data's class based on what it learned from previous data [11].

- **K-Nearest Neighbors** where the investigator assigns the number of neighbors k. A label point is chosen where it is the most common amongst the neighbors. The weight of contribution of that neighbor depends on how close it is [55].

- **Bayesian Networks (BNs)** is a directed statistical graphical model. It resembles a network where each node represents a random variable [10].

- **Formal Grammars** are different from these approaches. They consist of syntax rules

that describes the structure of the sentences in the domain. A grammatical parser attempts to parse each input using the inferred grammar. If successful, the input is accepted as part of the domain language. Graphical representation for grammatical parsing is usually done using a parse tree. Though formal grammars are very valuable and have several benefits, not a lot of investigators have used it in data mining. Therefore, this research is proposing a data mining approach by utilizing formal grammars.

2.3.1 Main Steps

To data mine, general steps are usually required. Most researchers breakdown the data mining process into six main steps [49][14][8][16][18]. A general outline of these steps are as follows:

1- **Understanding of the Application Domain:** Understanding the targeted domain, whether bioinformatics, business, or behavioral analysis, is essential before conducting any step. It requires understanding the problem to be solved and data mining objectives from field experts.

2- **Understanding Targeted Data:** Recognizing the nature of the data to be processed is critical in selecting a suitable approach to achieve expected results. In this step, the data is analyzed using a sample. Then we identify information about the source, the physical meaning of terminals in the language, and the strings structure that can be observed [5]. Knowing whether sequences are structured, semi-structured, or unstructured. Understanding how significant part of the sequence are identified if there are any relationships between different

parts. This will help in choosing the appropriate approach.

3- **Data Preparation:** Noise, repetitive, incomplete data has to be removed before building or applying the model. Skipping this step may result in inaccurate results.

4- **Modeling**: To build a model, first a training set is collecting that can represent all data types within the data set. Then constructing a model using the most suitable approach based on the type of data and the desired results. In this step, the information learnt in the previous steps is used to generate a grammar. However, when developing a model for a system, there are two main requirements to be considered [5]. First, the model must be an accurate representation of the target. Second, the model must be easier to manipulate than the original. In our dissertation, the focus is to model data's syntax combined and frequency.

5- **Evaluation**: Collecting a testing set from the data to evaluate the constructed model. The model is applied on the testing set. The investigator has to choose a suitable testing tool, such as the results accuracy or chai-square test, to evaluate his model. If the results were unsatisfactory, the investigator has to improve, or replace, the model to get better outcomes.

6- **Deployment:** When the data mining model generates satisfactory results, it can be deployed to used. However, further enhancements and updates may be done to improve performance as deemed necessary.

2.3.2 Challenges

Some common challenges arise when developing data mining algorithms [8] such as:

-Massive databases: Data storages are large and constantly getting larger. Data mining algorithms need to be efficient so handle Terabyte (10^{12} bytes), or maybe Petabyte (10^{15} bytes) within upcoming years.

-High data dimensionality: Dimensionality refers to the large number of attributes a single data input may have. Attributes can be distributed amongst different data bases.

-Continuously changing data: As new data is presented that belongs to the class, the class description may have to be updated to include new data.

-Overfitting: This problem occurs when the class definition is too constricted it will accept a small number of inputs. This also may result to a high number of false negatives.

-Structure of the Data: Data sequences are not always numbers and are frequently complex in structure. Sometimes, these kinds of data are ignored because of the structural characteristics. For this reason, grammars are deemed very valuable and needed. Grammars have the ability to describe more complex data structures beyond basic numbers and statistics.

-Syntax Modeling: The syntax of sequences are rarely static. For instance, the syntax of simple English sentence can range from a simple sentence with subject, verb, and object. To a longer sentence consisting of multiple adjectives and adverbs. This makes modeling these kind of data more challenging due to the varying length and dynamic structure. Grammars have the ability to describe structures with this type of modeling which makes it important in such fields.

2.4 Potential Applications of Data Mining Using Grammars / Probabilistic Grammars

It provides managers with important business intelligence critical for decision making such as fraud detection, marketing, and manufacturing [8]. It also helps researchers and scientists to understand certain phenomena and patterns. Data mining assists in making forecasts and predictions based on historical and real-time data. For instance, research can use data mining intelligence for learning future stock market trends and customers shopping behavior. Grammars are most valuable in areas when there is syntax. They have been used in a wide number of domains such as [27];

- Formal methods for process systems engineering.
- Formal methods for production chain management.
- Formalizing waste management.
- Formal methods for modeling biological regulatory network.
- Formal methods for specifying and analyzing complex software system.
- An algebraic approach to hardware compilation.
- Formal methods for UML.

Various valuable data mining algorithms are available in literature. However, very few works have been done using formal methods in data mining. There are two publications on applying formal methods for data analysis. A paper by Borges and Levene where they proposed a hypertext probabilistic grammar to learn from user navigation data [12]. In this paper, data on user's web navigation history is collected and stored. Based on the data collected, the paper proposes a way to extract information from the collected data set and learn user's web navigation pattern. Then,

probabilities are generated based on the analysis to predict the next page the user is going to. Their research algorithm is intended for real-time applications. The approach proposed in this paper is different as it is intended focused on static data.

2.5 Conclusion

The research's main objective and motivation of data mining using formal grammars is explained in this chapter. To better clarify main terminologies, key concepts were defined is also outlined in which includes formal grammars, probabilistic context free grammar, grammatical inference, and data mining. The chapter also lists the research approach and theoretical work in general. Each of the steps in this approach is further explained next in more detail in a dedicated chapter for each topic.

Chapter 3

Learning and Building a Model

3.1 Introduction

Before using formal methods to data mine, we will develop a structural model that describes the big data. The structural model will assist us in recognizing and understanding the structural pattern within the data. The process of developing such structure is called grammatical inference [26][6]. Theoretically, a grammar can be inferred for any structured data. A model is built based on the inferred structure to analyze the patterns, probabilities of each re-write rule in the grammar, structure, and relationships within sequences belonging to that class of patterns. Having various classes of patterns mean a grammar has to be inferred to describe each class. These generated grammars can be combined into a composite grammar [5].

An inferred grammar describes large set of data by creating rules that govern complex relationships between sub-sequences and the overall structure of that data. This makes formal methods (formal grammars) serve as a descriptive model for the large data set in a condense matter. It is also able to describe the correlational dependences and relationships within the structure or sequential pattern [24]. In addition, the ability of formal methods to describe data structures without location dependency features. This makes it more flexible to describing various sequence length from the same family. A parser can be built based on that

grammar that can associate sequences to a certain class of patterns. Grammars can be used in various applications such as help identifying errors, like detecting a missing verb in an English sentence. Another example is help predicting future results such as studying the buying behavior of customers and predicting what will be bought on next visit based on purchasing patterns. These are just some examples as which field this technique can be utilized.

In order to learn a structural model, first a grammar for the targeted class must be inferred. To infer a grammar, in general, three main steps are followed [5]:

1- Identify and analyze targeted data. A sufficient sample for a training set, also called a positive sample, is collected. The collected data sample is used to represent the overall large data set. The investigator then analyzes the data structure for the strings and the relationships among substrings.

2- Choose the suitable grammar type based on the data's nature. There are several types of formal grammars [17]. Choosing the grammar type depends on the natural of the targeted data. Build grammar by analyzing the collected training set. Then, extract the grammar's start symbol, rules, terminal symbols, and non-terminal symbols.

3-A different testing data set is collected to validate the grammar's fit. If the testing results are satisfactory, the grammar may be deployed (Figure 3.1).

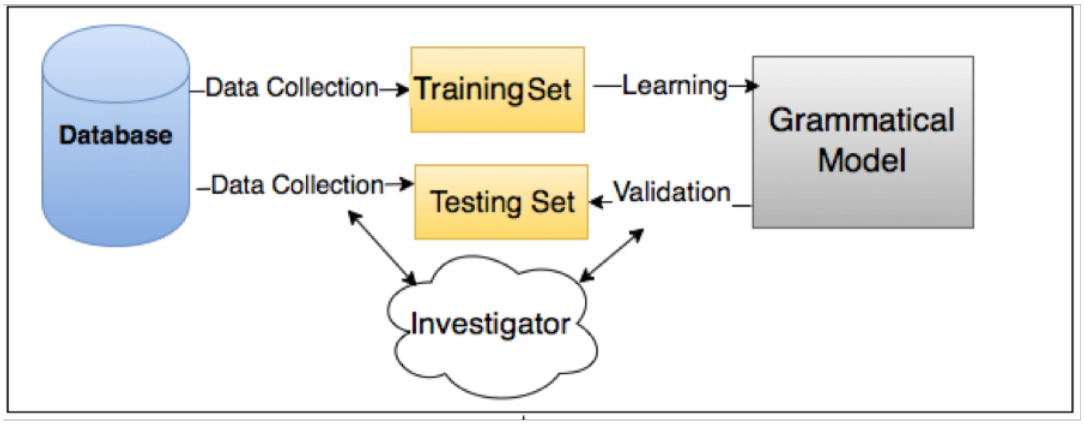


Figure 3.1: Building a Structural Model for a Database by Using Grammatical Inference

Learning and building a model is the first stage of this dissertation (Figure 3.2). Extracting a structural model serves as the first stage and the bases for mining big data using formal methods. It acts as description for a family class that is contained in the big data. A single model is built for each family class within the big data using formal methods as shown in figure 3.2 below.

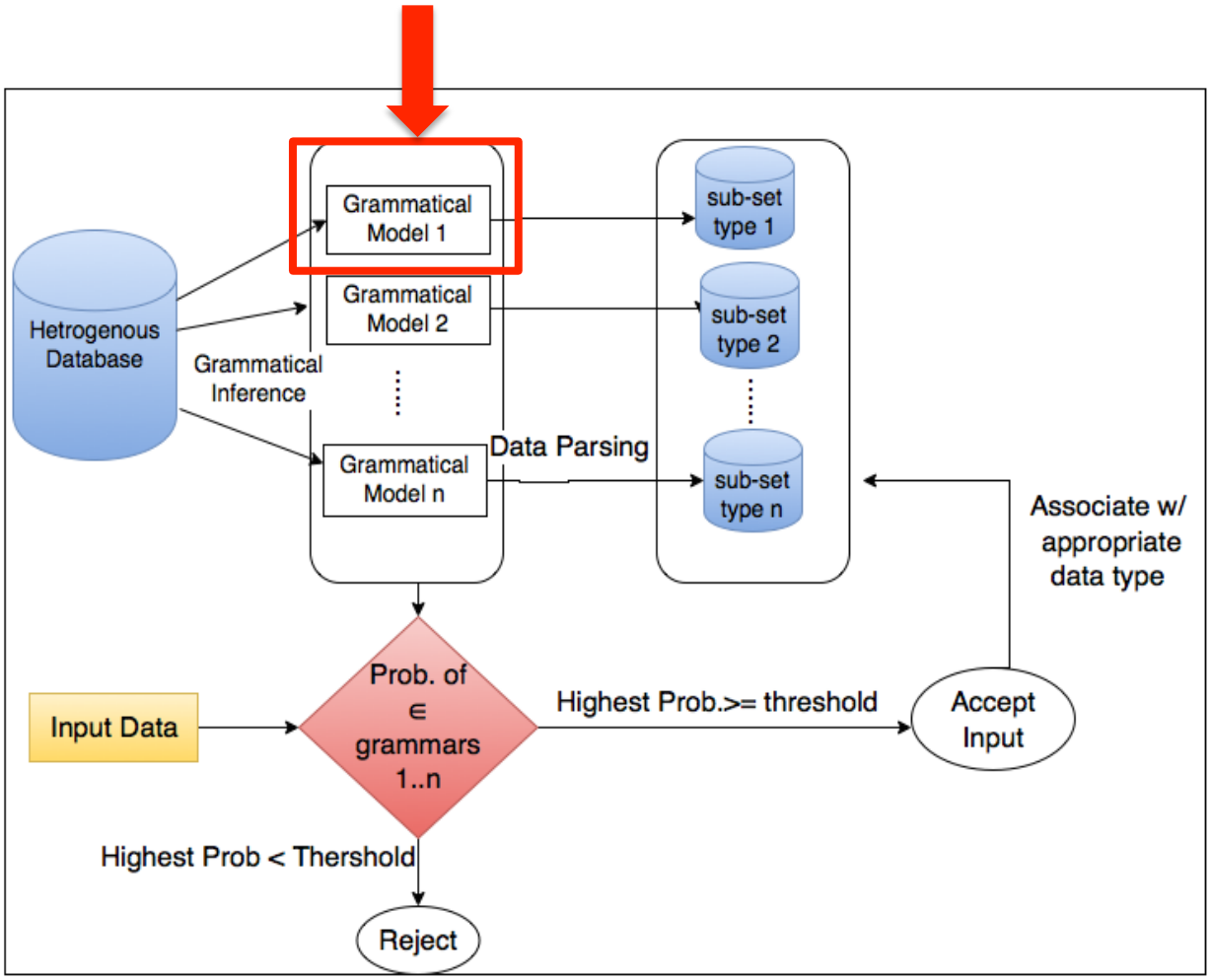


Figure 3.2: Learning and Building a Model Stage in Overall Dissertation Roadmap

3.2 Challenges

Unsupervised grammatical inference is still and open problem. The two main issues discussed in inference are over-fitting and over-generalization [31][9]. In the over-fitting problem, the grammar represents one exact sequence only. Any variation of the sequence will be rejected. When inferring the structural description for a specific family class, inferring one for each sequence in the family is absurd, especially if the class contains thousands of strings. The model needs to accept a set of sequences that belong to the correct definition of the class. An example of a

grammatical inference tool is Sequitur [47]. The algorithm accepts a biological sequence and generates the context free grammar for that sequence. However, it can only generate a grammatical inference for a single sequence. This raises the problem of over-fitting.

In the opposite side of the spectrum is the problem of over-generalization. The problem occurs when the class structural description is broad that it accepts sequences that are not part of the data set. The investigator has to find a balanced description that best describes the targeted class without being too restricted or too broad.

Various solutions were proposed [5][31][9] to overcome these issues. Gold's theorem [31] explains how it is impossible to infer a class's characteristic that represents infinite sentences with positive sample only. Gold explains that there must be some restriction to represent sequences that do not belong to the class such as a negative sample to avoid over generalization. This is done using negative sample in addition to the positive sample for inference. However, this is not always feasible as negative samples are not available in some applications. Angluin proposed another solution using an interactive inference process[9], such as a teacher. This requires building a model from positive examples. Then the model attempts to generate more samples that belong to that class. A human teacher has to provide feedback whether the generated data from model is part of the class or not. The model then modifies the learnt structure accordingly until it gets high positive feedback from the data generated. However, using this method does not

apply in all fields. For instance, in the case of DNA like in our experiment, it is hard for a teacher to tell if a DNA sequence generated by the grammar is part of a particular domain by simply looking at it. Therefore, a tool, called Protomata-Learner [20][19], was chosen to infer data's structure which utilizes finite state automata for inference. It overcomes the over-fitting and overgeneralization problems by extracting structural information by conducting characterization step, which involves identifying and ordering partial local multiple alignments. Then applying generalization, which merges similar partial local multiple alignments for a more general representation of the sequences.

3.3 Related work

Research in field of predicting the structural features of a sequence belonging to a class has been an attractive field in various fields in general, such as linguistics [42], bioinformatics [53], and behavioral analysis [6]. Various proposed valuable methods have been proposed for structural class predication. One of the many fields that require predicting the class's structure is bioinformatics for different representations and predictions of RNAs, DNAs, and Proteins. Xiao, Wang, and Chou [62] published a paper explaining a unique approach where they represent the structure of proteins using by converting it into an image. It converts the amino acid symbols into a binary representation of 0s and 1s. These binary representations are then converted into a gray image where 0 is white and 1 is black. It assumes that proteins with similar texture on the image generated belong to the same structural class. The approach resulted in a good overall class prediction accuracy. However, if a certain common

substructure in a class might need further investigation and study. It can be challenging to identify the interesting region by looking at the image alone. The investigator will have to get a sample of the class and try to link the part it needs studying from the image with the class sample. In this research the common symbols and motifs are clear and easy identified.

Bystroff and Krogh [13] used Hidden Markov Models (HMM) to produce protein's structural model. They use the protein's amino acids as input symbols and the structural feature as the output. The transition states define the topology of the overall structural model. It calculate the probability of amino acids transitions. Our work is different where it represents structural model using formal methods, which has more ability to represent the embedded correlations in sequences.

3.4 Building a Formal Model

The main objective is to build a model that recognizes the statistical and structural patterns within a large set of data using formal grammars. In order to achieve this goal, various steps must be done. This research will learn formal grammars from training data with known classifications. Afterwards, the learnt grammars will then be used to classify a large set of unknown data points. There are three phases in our approach. First, inferring a grammar from a sufficient training data that represents the class. Then, building a data mining (grammar based) model. After the model is built, it must be validated. A different set of data is collected, called testing set, other than the previous training set. The sample is run on the parser. Sequences from the training set will be accepted or rejected as part of the data set. If

the model results are unsatisfactory, a new grammar is inferred. Finally, when the model results are satisfactory, the model can be deployed.

The training data is first collected to represents the targeted data set. It is then run through inference tool to produce a graphical representation of the structure. The structural representation is then encoded into a parser. Each substring region with significant structural characteristics it can be seen as a structure with a grammatical model by itself. These structural regions are concatenated to produce the overall string (the composite model) (Figure 3.3).

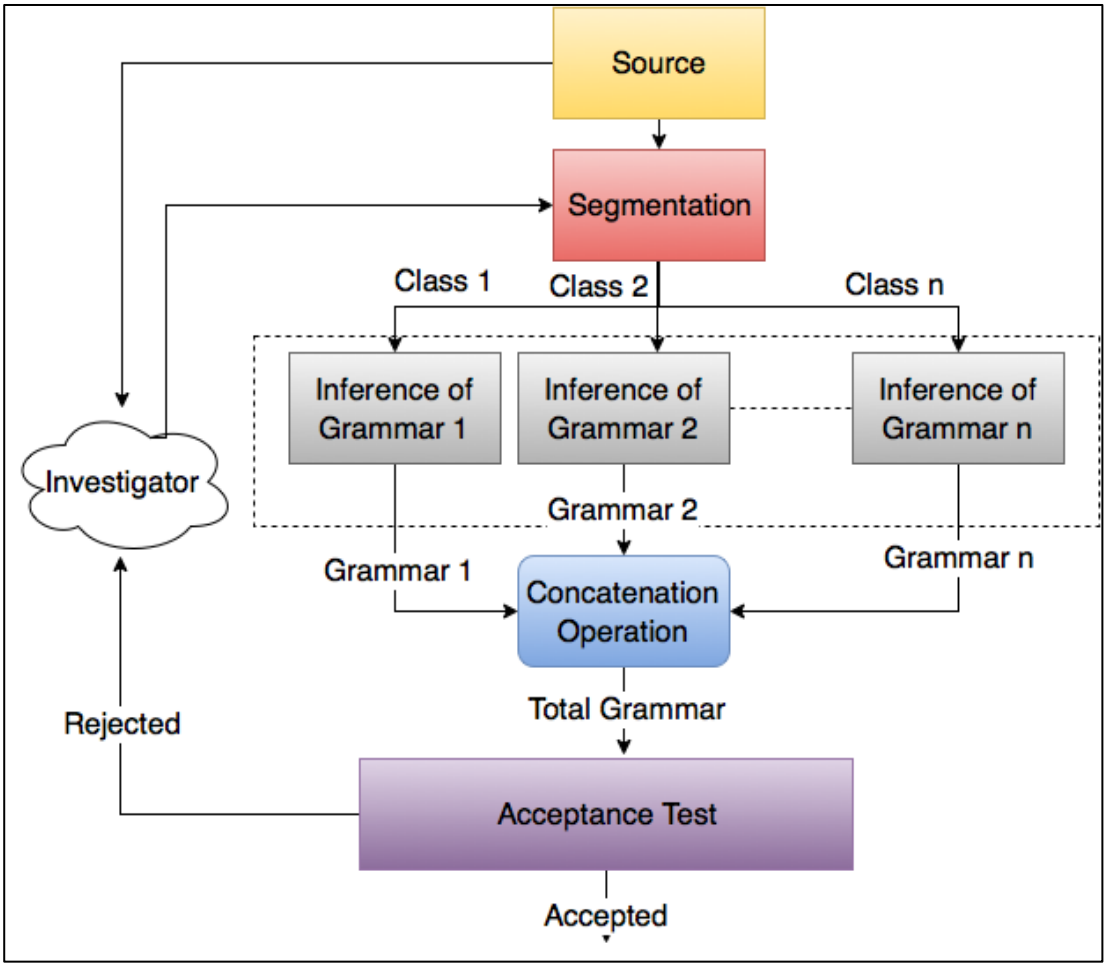


Figure 3.3: Inference for Complex Sequences [5]

The concatenation operation of *a* and *b* is a binary function written as *ab* [56]. As for language concatenation, let L_1 and L_2 are two languages. The concatenation of the two languages L_1 and L_2 will be [5]:

$L_1L_2 = \{ s_1s_2 \mid s_1 \in L_1, s_2 \in L_2 \}$

Having s set of languages L1, L2, L3, …., Ln where the pre-set length of the sequence is m >=n, concatenation can be used to form a new language [5]. For instance, we can use the set of languages to form a new language L = L1 L3 L1 L2. If we have some S that is a subset of L ($S \subset L$), then we can divide string in S into m sub-parts. Each part belongs to one and only one of the language sets. Concatenating two regular languages (finite state) will change the grammar to become a new higher state grammar. Also, one must take into consideration that the order of these sub-parts are in the same concatenation order of the languages they belong to [5]. For instance, lets assume we have some language L = L1 L2 L1 and some string t that belongs to the language L. Then, t can be divided into three part t = {t1, t2, and t3} where each of the parts has the length of m. Then, t1 and t3 must belong to L1 (t1, t3 $\in$ L1). Also, t2 belongs to L2, (t2 $\in$ L2). L1 has the grammar G1 with starting symbol S1 and terminal set {x1, x2, …,xn}. Also, L2 has grammar G2 with starting symbol S2 and terminal set {y1, y2, …, yz}. The new language L resulted from concatenating L1 and L2 will be as follows:

- L(G) = L1(G1)L2(G2) ,
- The new starting rule will be S → S1S2 (figure 3.4)

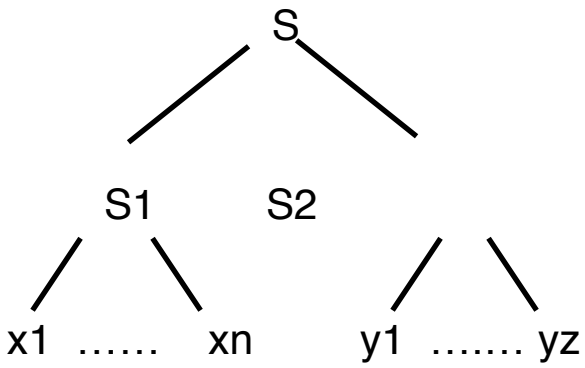


Figure 3.4: Parsing tree for a new language L(G) from concatenating L1(G1) and L2(G2)

However, we must note that concatenation should not be confused with the union function (U). The union of the two languages L1 and L2 will result the new starting rules:

S→ S1

S→ S2

Which is different than the concatenation starting rule S → S1S2 .

3.5 Experimentation

For the purpose of experimentation, data was collected from an online repository for protein sequences UniProt http://www.uniprot.org [60]. It was collected from the

UnoProt's Knowledgebase (UniProtKB) section. UniProtKB is an accurate database containing a large number of real protein sequences of various species such as eukaryote, bacteria, viruses, and archaea. Viral nucleic acid binding protein of different virus types (Garlic common latent, Butterbur mosaic, Blueberry scorch , etc..) were used to conduct experiments. Our aim is to test our approach by automatically identify nucleic acid binding regions based on structural characteristics. All sequences were in FASTA format (A text-based format used to represent nucleotide or peptide sequences[63]). Data were stored and handled in text files.

A. Data Preparation

Before using the collected data, it must be prepared and cleaned. This is done in two steps:

1- First, all repeated and redundant sequences where removed from the sample.
2- Then, all sequences that are incomplete or had invalid characters that do not belong to the amino acid characters were omitted. By invalid characters we mean the symbols that are usually added to protein sequences to represent an unknown base represented by (X). Also, characters that are not part of the amino acid symbols such as J, O, and U.

After the collected data was prepared, the total viral nucleic acid binding protein sequences were 252 sequences. Data was randomly divided into a training set to learn and build the grammar and a testing set to validate and test the model built using k-fold cross validation method [36]. Here, k is set to 3. In 3-fold validation

method, the total data is split into three parts (Figure 3.5). Here, 3 folds are deemed number to give a good estimate of precision with a good size of training to describe the sample and a training to test the generated structure. With the 3-fold, each training set is composed of %66.777 and the testing set is composed of %33.333 of the total data set. Each testing data set in this experiment consists of 84 sequences. The training data set contained 168 sequences. Testing in this experimented was repeated three times while changing the training and testing sets for each fold.

Here, the three exclusive parts were used so that one part is used for training and the other two parts are used for testing and validation. Then, the old training set is returned to the total data and another set is used for training. We do this until all three sets are used for training. In other words, if testing set x is used for testing in fold x, where:

Training set for fold x = Total collected data - testing set x

Testing set1 +Testing set2+Testing set 3= Total data

Testing set1 ∩Testing set2 ∩Testing set 3= ∅

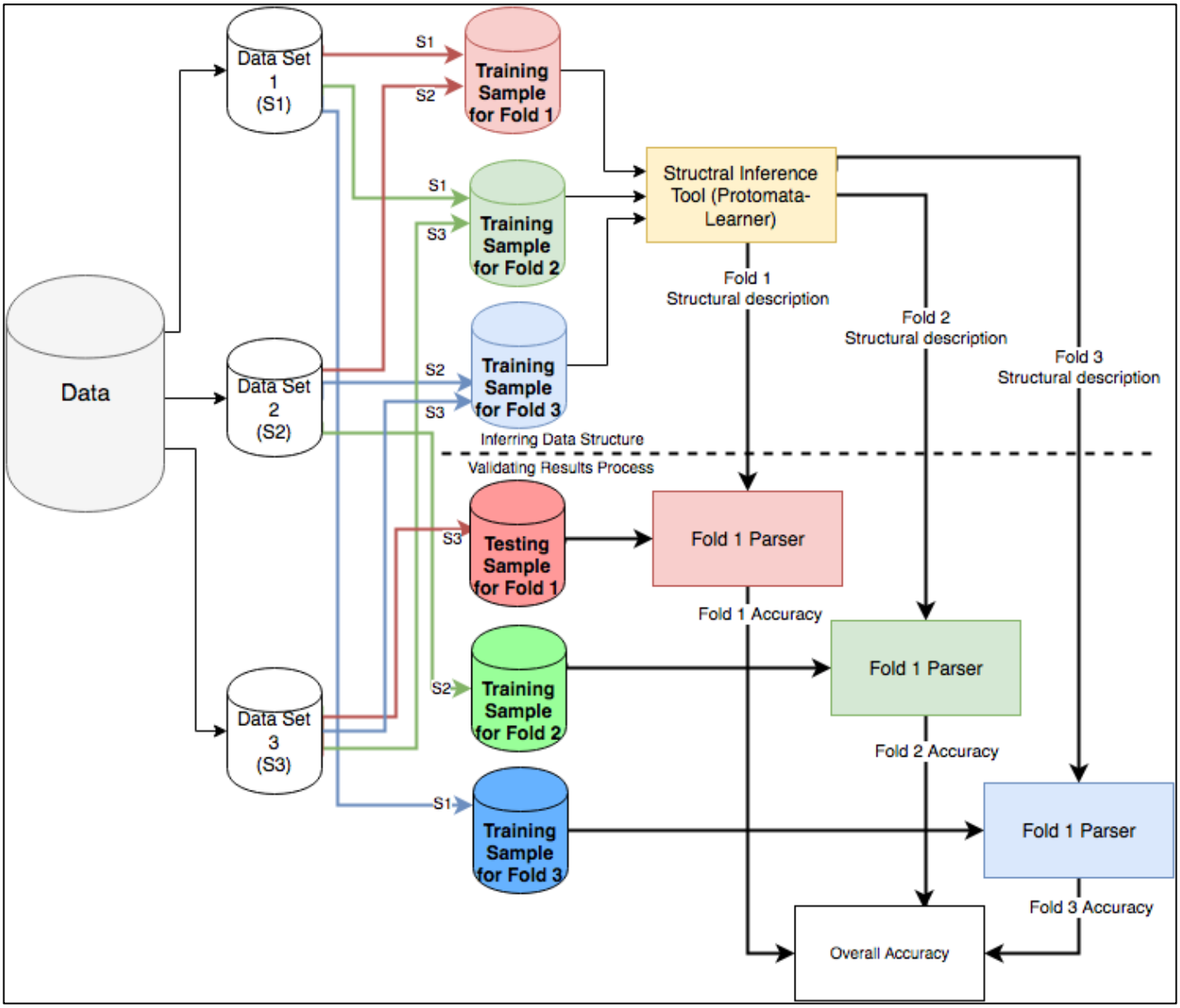


Figure 3.5: Structural Inference and Validation Using the 3-Fold Cross Validation Method

B. Building a Model

In each fold the training set is fed into the tool (Protomata-Learner) [20] [19]. Then, the tool generates a graphical structure representation. In figure 3.6, we can see part of a structural representation from our experiment. The generated graphic from Protomata highlights the common repeated sub-sequences from the training set. Protomata utilizes finite state automata [20][19] for infer the class overall structure. These repeated sub-sequences are used to identify the sequence's overall structure. The graphical structure helps to understand the overall family class. The overall generated structure of this experiment can be found in attached appendix 1 at the end of this dissertation. Here, the terminal symbols consists of the 20 amino letters notations {A, R, N, D, C, E, Q, G, H, I, L, K, M, F, P, S, T, W, Y, V}.

The structure of the family class obtained from the training sample. Then, we encoded the generated structure and built a parser in C++. We fed the sequences from the testing sample into the parser. We repeated the previous steps three times for each of the 3 folds. Different new training and testing sets are collected for each fold as explained in previous section.

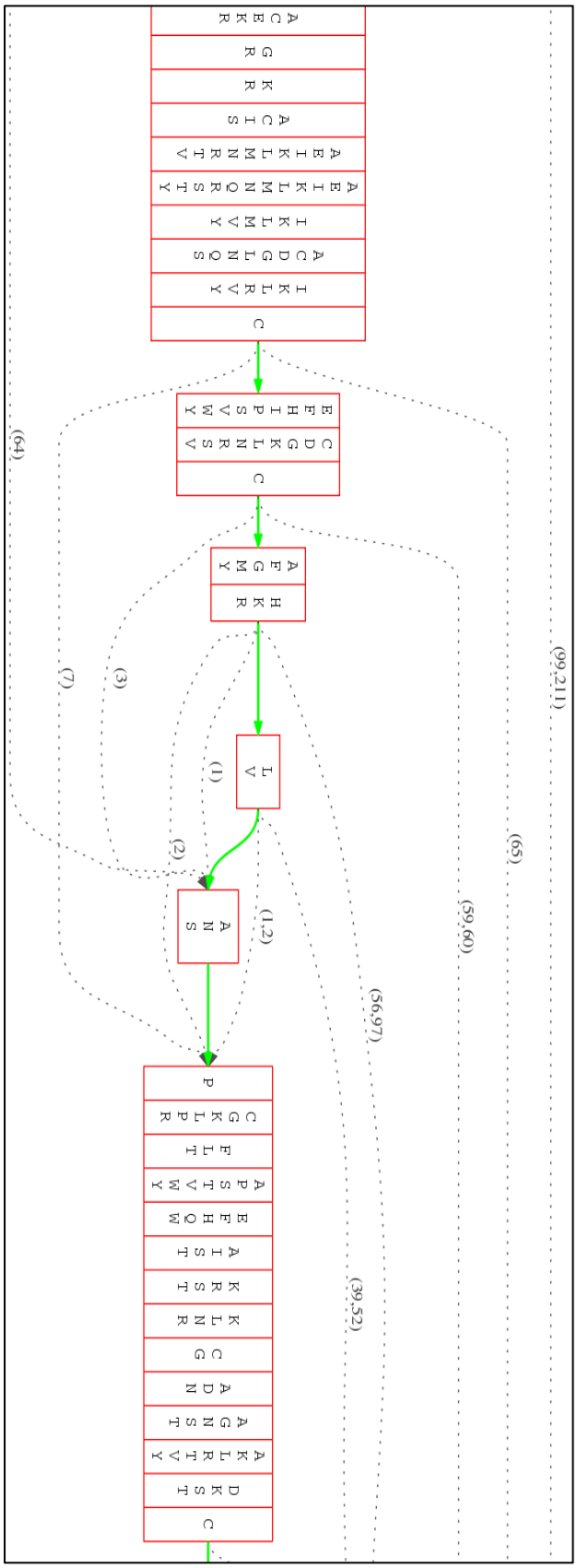


Figure 3.6: Part of a Structural Graphical representation of Viral nucleic acid Binding Proteins Class from Protomata-Learner

C. Validation

Next, the parser needs to be validated. The testing set is used to test if the structure was able represent the structure of the data through the parser. As mention previously, a 3-fold cross validation method. Here, we have 252 overall sequences. This means 168 were used for training and fed in to the tool to infer the structural characteristics and then build the parser. Then, the remaining 84 sequences were used for testing it. Since we have three folds in this experiment, there will be three parsers. Each one encoded using one of the training samples. When run, the parser will accept the data sequence from the testing set if the program is able to parse it (i.e. match the generated structural model). After all sequences in the testing set are validated, the model is claimed that it is the correct. However, if the majority of data sequences were unsatisfactory (could not be parsed using the inferred structure), we need to infer a new structure and repeat the inference process (Figure 3.7).

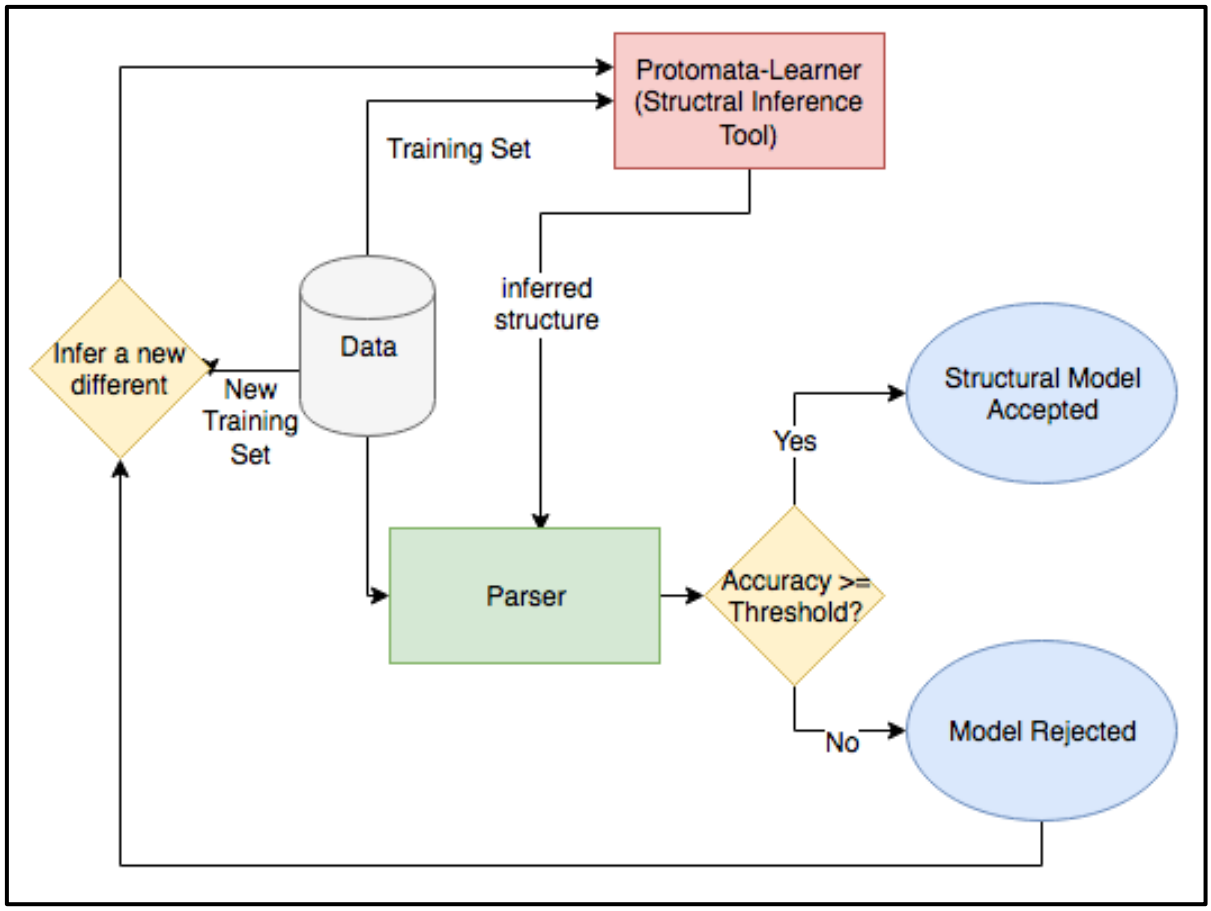


Figure 3.7: Inferring A Structural Description Process

3.6 Results and Discussion

To validate the approach, each of the testing sets is used with the associated encoded parser. After all sequences in the testing set are parsed, it calculates the number of the sequences that are accepted as part of the structure. It will then be identified as part of the family class based on its structural features.

In this experiment, we tested it using viral nucleic acid binding protein sequences. Therefore, accuracy is then calculated by counting the number of sequences in the testing data set that were accepted by the parser from the testing set (i.e. being part

of the structure inferred from the training set). Repeat the process three times for each fold and calculate average accuracy results. In the first fold, 70 of the 84 sequences in the testing sample were accepted. In the second fold, the number of accepted sequences went higher to reach 81 sequences out of the 84. The last third fold resulted in 78 accepted sequences (Table 3.1). The total average of identified sequences from the total sample was %90.87 from the three folds. The recall percentage was calculated as:

$$\text{Recall} = \frac{True\ Positives}{Total\ Sample}$$

Fold #	Accepted	Rejected	Recall Percentage
Fold 1	70	14	%83.33
Fold 2	81	3	%96.42
Fold 3	78	6	%92.85
Total	**229**	**23**	**%90.87**

Table 3.1: Approach validation using a 3-fold validation method

Various researchers in biology tried to capture the difference and similarity among biological sequences of the same class. Traditionally, it was studied using consensus sequences methods through multiple sequence alignment. To incorporate some structural difference within sequences of the same family, insertion and deletion was then used [21]. The approach was easy and straightforward [48]. However, this approach does not provide information about location variation, does not provide

scoring or probability, and requires alignment of all sequences involved [21]. Therefore, the study of sequence structure and pattern detection was an interesting field for researchers. Regular expressions (regex) was then employed to identify common patterns within sequences such as Emotif-Maker [46] and Pratt [34]. They are fast and easy to implement especially on small pattern sequences. Then again, small sequences identification may lead to high false positives. It is also more challenging to identify long sequences with high variations in symbols correctly. Some of the latest approaches in literature that studies the structure of biological sequence that focuses on pattern recognition are Teiresias [52] and Splash [15]. However, these methods' expressive powers are all below the simplest level of formal grammars, regular grammar [21]. They are not able to extract correlations between certain positions. When comparing Protomata-Learner, which uses regular grammar, to other valuable pattern recognition methods such Pratt and Teiresias, we see that it preforms equal or better results [19]. Pratt, Teiresias, and Protomata-Learner resulted in 78%, 89%, and 87% accordingly. Also they resulted in 90%, 23%, and 100% in precision. This shows how formal grammars have powerful structural expression capability.

3.7 Contribution

In this work, we have shown how inference using formal methods can be utilized to build a structural model to describe a class of patterns. Formal methods are very valuable in describing structures and patterns. In addition, formal methods have the advantage of identifying correlations within parts the structure. This model is a first step and serves as a base for data mining big data using formal methods. It also

helps understand the overall structural characteristics of a family class. These model can also provides statistical and structural representation of data in a condense matter.

3.8 Conclusion

This chapter shows an approach for inferring a structural model using formal methods in data mining. This model describes structural patterns within a big data for a certain class. It can serve as the first step in data mining using formal methods. As data mining using formal methods have various benefits such as describing a structure for a class family in a condense matter. Help discover new relationships between sub-sequences within the pattern. Also, formal methods has the benefit of being able to overcome location-based recognition. Viral nucleic acid binding protein sequences were used for experimentation in this research. Also, Protomata-Learner was used as the inference tool. First, data sample for training is collected and cleaned from incomplete and redundant data. Then, using the training, the structure of the data is inferred. A composite structural model is built. This composite model represents the given big data class. The generated model is then validated using a k-fold cross validation method to make sure that the composite model is accurate enough for representing the given big data.

Chapter 4

Data Association

4.1 Introduction

With the advancement of data collection and storage, large volumes of heterogeneous data are being collected. There is a need to associate each data instance to the correct class automatically. Researchers usually need to study a certain class from within this large set of data. The previous chapter discussed how to describe a deterministic family class's structure using formal methods (formal grammars). This chapter will explain, given a structural model, how to associate complex sequences to that family class using probabilistic grammars (Figure 4.1). This can be challenging especially with classes with complex structures and simple regular grammar will not be enough for this type of data. Data association in this research is done using probabilistic context free grammar (PCFG). Probabilistic grammar in this research is inferred through interactive grammatical inference. First, the general structure is inferred using the previous used tool, Protomata-learner, which utilizes regular grammar. Then, the grammar is later improved by input from the investigator. Formal grammars are a very valuable tool to describe and identify complex sequences as mentioned in previous chapters.

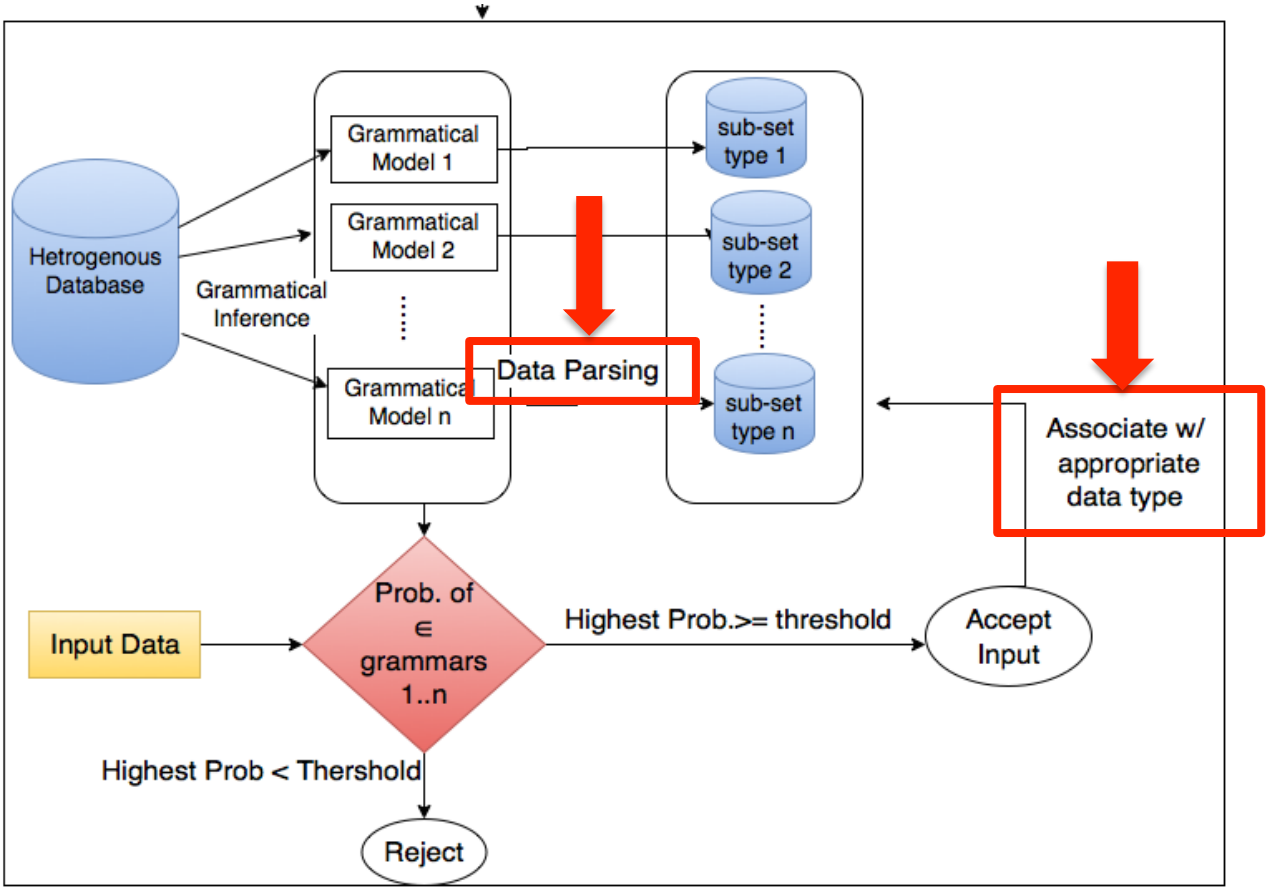


Figure 4.1: The overall framework of the dissertation with highlighted parts of this chapter's topic focus

To parse sequences using grammars, we first built a structural parser. The first approach we used the same tool used in precious chapter, Protomata-Learne, to learn the structure of the class. However, the tool was not sufficient to describe the structural inter-relation between sub-regions important for structural identification. Afterwards, an attempt was done by encoding the standard description of the class into a CYK parser algorithm [35]. However, the parser is able to integrate the ability to describe symbol variation and location based identification. Therefore, we are extending the grammar by applying on top of our grammar an error correcting grammar algorithm by Rajasekaran and Nicolae in 2014 [50] to be able to describe

complex sequences. Then, to further improve our results, we modify the algorithm to replace the error count with probabilities. For each input data sequence, the parser generates the probability of that sequence being part of the structure. If the probability is less than a certain pre-set threshold, the grammatical parser will not accept the sequence. In this research, E. Coli Promoter sequences are used for experimentation. In section II, the main challenges of this research is addressed. Then, section III presents some other work done related to this paper. The overall data association approach is explained in details in section IV. An experiment is then demonstrated for this approach in section V. Section VI presents experimentation results. Finally, the chapter is concluded in section VII.

4.2 Challenges

Associating a data sequence can be straightforward for simple structured sequences. However, various data patterns have complex structure that cannot simply be identified to a certain data pattern instantly. For instance, some sub-structural characteristics can depend on the existence of other sub-structures within that input sequence. Also, a certain sub-structure can appear in various formats. These issues make associating the data sequence based on its pattern challenging. Here, the paper presents in later sections an approach that aims on overcoming these issues.

4.3 Related Work

Leung, Mellish, and Robertson [38] present a grammatical parser to model and predict DNA transcription binding sites using Definite Clause Grammar (DCG). The parser takes the string and parses it to conclude if there is a transcription binding site. However, DCG can only be implemented in Prolog specifically. On contrary, this research parses DNA transcription binding sites using Context-Free Grammar (CFG). This methodology can be implemented using any common programming language and is not limited to a certain one.

Another related approach by Krogh, Mian and Haussler [37] where it uses Hidden Markov Model (HMM) to predict protein coding genes in E. Coli DNA sequence. Their approach combines several methods of inspecting sequences for initiation signals, scoring of potential coding regions, and dynamic programming using an HMM framework model. Their approach identifies %80 of previously known E. Coli DNA sequences correctly. Our research on predicting E. Coli DNA sequences using probabilistic context-free grammar yields a higher prediction result of %96.226.

4.4 Associating Data Using Grammatical Models

An input sequence can be associated to a certain data pattern structure family if its parser accepts the sequence. In this research, we build a model that parses sequences using the grammatical description of the family class. First, the general grammar is inferred using a tool. However, some structures might need further enhancement. This can be done by using a negative sample in addition to the

positive training sample [9]. Also, a human expert, or teacher, can also be used to provide feedback on the inferred sample. The second method is called interactive inference, which will also be used in this research.

After the grammar is inferred, a parser will analyze and model a given sequence to a grammatical structure. If the parser was able to match the sequence to the structure successfully, the grammar will accept the input (Figure 4.2). Some simple structures can be inferred and described using regular or context-free grammars. Then, a parser is then built using one of the parsing algorithms available in literature such as Cocke-Kasami-Younger (CKY) and Earley parser [35].

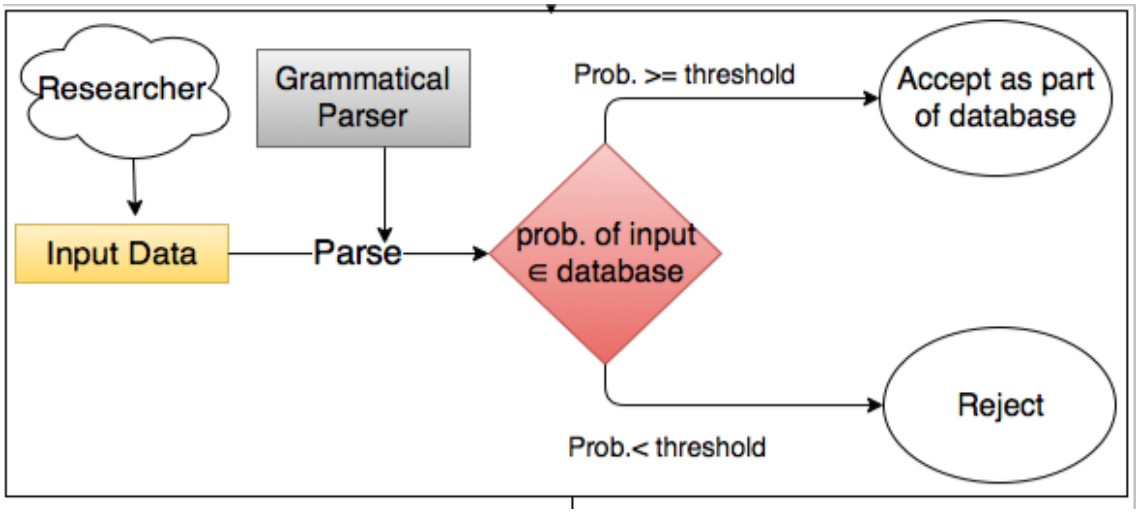


Figure 4.2: Associating Data inputs to Classes using Probabilistic Grammatical Models

However, in some research fields, data sequences are more complex and require additional steps to simple grammars to model. Biological sequences such as DNAs are an example of such sequences. DNA sequences are identified by motifs which

are sub-sequences. Motifs are important components for identifying the type of species and region of that DNA. In some structures, motifs symbols are not fixed. For instance, the TATA-box motif region is found as “TATAAT” in it’s standard form. However, other variations (such as “TAAAAT” or “TATACT”) are also considered a TATA-Box. Also, the location of the TATA-box in comparison to other motif regions the string is significant in identifying the region. Therefore, a more powerful grammar is required to incorporate these features. In the next section, we will see how an existing error correcting grammar [50] was used in this research to incorporated to be able to describe the structure of E. Coli Promoter regions. Then, it was further improved by modifying it to replace error count with probabilities.

4.5 Experiment

For experimentation, Escherichia Coli (E. Coli) Promoter DNA sequences were chosen. DNA sequences in general comprise of a sequence of 4 nitrogenous reputations {a, g, c, t}. However, certain patterns have specific meaning and significance. Since there are various types of DNA species and types with each having its own structure and implications, the investigation will concentrate on E. Coli DNA promoter sequences. E. Coli species are bacteria that can be found in the environment, food, and intestines of humans and animals. The nature of biological sequences, such as DNAs, are complex. Figure 4.3 below illustrates the E. Coli promoter sequence in the standard form.

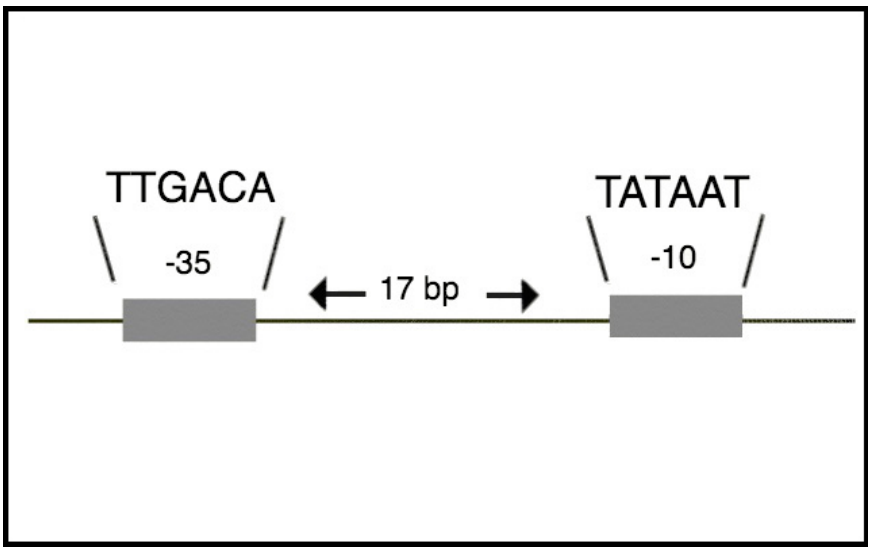


Figure 4.3: A Standard E. Coli Promoter Sequence Structure

There are three significant parts to identify an E. Coli promoter sequence:

- TTGACA which is located in the -35 bp
- 17 base pair middle sub-sequence: found between the TTGACA and TATAAT regions.
- A TATAAT sequence (also called TATA-box). Which is located in the -15bp location.

A) Inferring a Grammar

In order to build a parser to associate sequences to the family class, we need to infer the grammar. A sample of 106 E Coli DNA promoter sequences is collected from the Machine Learning Repository UCI [39]. It contains a set of 53 positive and a set 53 negative. The positive set was used for inference and fed to Protomata-Learner tool to

learn the general structure of the sequence. A structural graph was generated in figure 4.4. In the graph generated by Protomata-learner, we can see that it was able to identify two important regions for E. Coli promoter DNAs. However, the identification of the two region is not enough. The length between them is also an important part of the structural identification. If the regions are too close or too far the sequence is not of the structure class. This will lead to a number of false positives. Also, not a lot of variation in amino acids are incorporated in this definition. This will lead to false negatives.

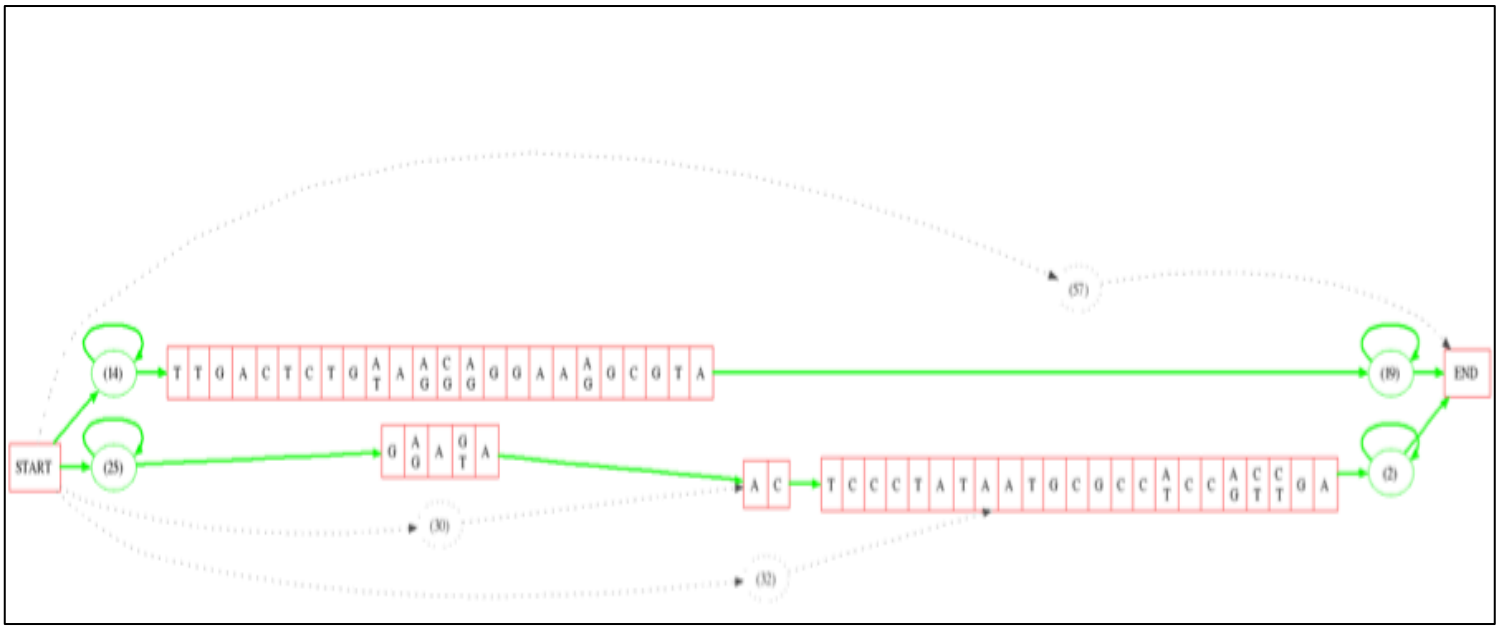


Figure 4.4: General Structural Graph of E. Coli Promoter Sequences Generated by Protomata-Learner

B) A Context-Free Grammar for the Standard Form

In this research, an approach is proposed that allows Context-Free grammars (CFG) to be applied on E. Coli DNA promoter sequences. CFG was able to parse E.

Coli Promoter regions in their standard form (Figure 4.5).

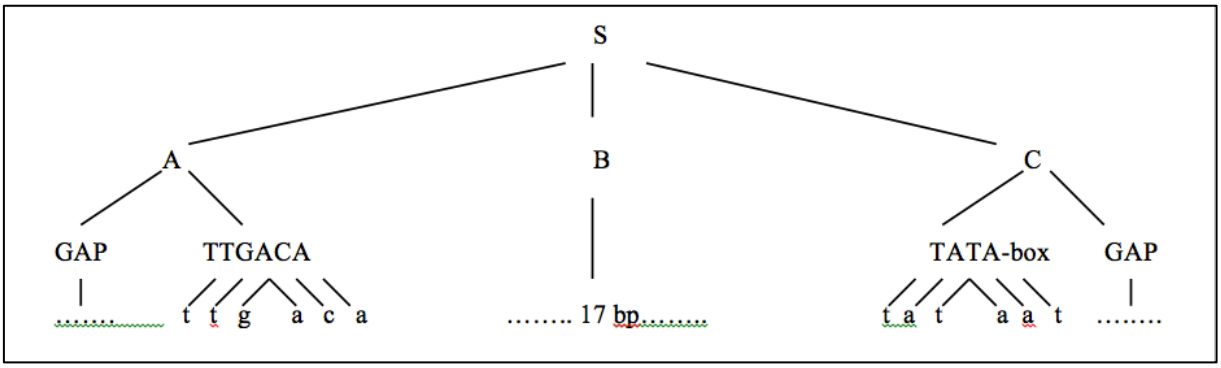


Figure 4.5 : Parsing Tree for an E. Coli Promoter DNA Sequence in the Standard Form

When the standard form grammar was tested on real data, few sequences were accepted causing high false negatives. E. Coli promoter sequences are rarely found in the standard form. The challenge is to identify several sequence variations when parsing and associating it to the correct category (e.g. How much difference from standard form should be accepted?). Also, DNA sequences have variety of lengths. Identifying a significant region based on location will yield incorrect results. Another challenge is differentiating between a gap and a significant structural region. For instance, in the TTGACA region, a TAGCCA and CTAACA can be considered a valid TTGACA sequence. Also, the mid- sequence can range from 15-19 bp rather than the 17 bp length and still be considered valid. As an example, the following is a random E. Coli promoter sequence from UCI Machine Learning Repository (Promoter Gene Sequence) data set [39].

tatcctgacag **ttgtca** cgctgattggtgtcgt **tacaat** ctaacgcatcgc

The TTGACA and TATA regions are the ones in bold. The sequence of symbols

between them is the mid-section. In the first region, we see (TTGTCA) where in the standard form it is (TTGACA). This one symbol difference from the standard form will result to that standard parser to reject the sequences as an E. Coli promoter. The distance variation needs to be accepted till a certain threshold. For instance, in the previous example, the sequence is a valid E. Coli promoter that doesn't match the standard form. After comparing all significant regions in this sequence, this sequence has a distance of 3 from the standard sequence.

C) Applying Error Correcting Cover Grammar

Due to the nature of DNA sequences, there was a need to propose an extension to CFG in order to be able to express these strings. The challenge is to identify how much distance is considered valid till that sequence is rejected as an E. Coli promoter sequence. Also, how can these differences described using Context-Free Grammars. Therefore, there is a need to extend Context-Free Grammars by using an error correcting parser [50].

The error correcting cover grammar was proposed by Rajasekaran and Nicolae that can parse in a cubic time (2014) [50]. From the developed standard E. Coli Promoter grammar, necessary adjustments are made as explained in their paper [50] for applying a cover grammar. The cover grammar Ğ contains the same terminal symbols and start symbol as in the original grammar G. However, it has some added production rules and non-terminal symbols. The grammar will be Ğ= <T, Ň, Ř, S >. A symbol will be added on top of all the production rules in P e.g. (A □ B C). It will represent the

error value associated with that rule.

All the production rules in Ř will have an ℓ with value of 0 since there is no errors in these rules. To incorporate errors, for each non-terminal A ∈ N where:

A □ b ∉ R we add A–^{1}□b ∈ Ř ∀ b ∈ T where b ≠a.

The parser counts each distance of a given sequence from the standard form. Parsing the sequence where the number of errors is lowest. Each difference from the standard form is considered an "error" or one point distance. If the distance exceeds a specific threshold, the sequence is rejected and said not to be an E. Coli promoter sequence.

D) Adding Probabilities

To improve the model's accuracy, probabilities are used. Probabilities of the model's rewrite rules are calculated in being in a certain location in the region replaces the error count. The approach is extended and modified to calculate probabilities instead of counting distance. A biomedical paper by Lisser and Marglit [40], where they analyze E. Coli promoter sequences and calculate probabilities of each symbol being an a, c, t, or g in each location of the TTGACA and TATAAT on 300 E. Coli Promoter sequences (Table 4.1). The average probability of accepted sequences gives an insight on how well the grammar represents the targeted data. If the total probability of the sequence P(x) is less than a certain threshold, the sequence will be rejected.

$$P(x) = \begin{cases} x \notin data\ set, & P(x) < threshold \\ x \in data\ set, & P(x) \geq threshold \end{cases}$$

P(x) is the probability of the string x. To generate x, the rewrite rule r is used m_r times

[49]:

$$P(x)= \prod_{r} (P_r)^{mr}$$

where $0 \leq P_r \leq 1$ and P_r is the probability of rewrite rule r.

The following is a random E. Coli promoter sequence from UCI Machine Learning Repository (Promoter Gene Sequence) data set [39] as an example.

tactagcaatacgcttgcgttcggtggttaagtatgtataatgcgcgggcttgtcg

As we can see, it is hard to identify significant regions by just looking at the DNA sequence. Therefore, a grammar is going to be inferred for each region (Gap, TTGACA, Middle, and TATA-box). Then, each region is going to be joined with the previous one using concatenation to generate the overall structure (Figure 4.6). Concatenation is implemented so that the overall language is equal to the concatenation of the language of the first grammar G1 and second language of grammar G2. L= L1 L2 = G(L1) G(L2) [5] .

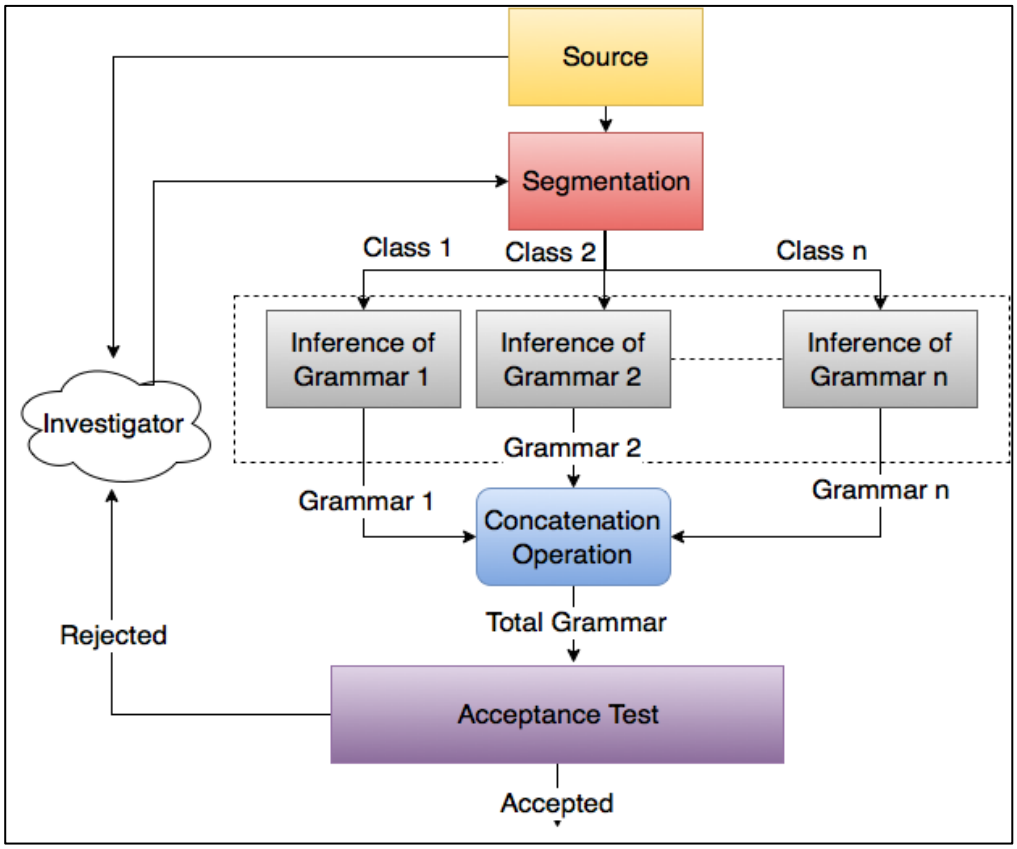


Figure 4.6: Inference for Complex Sequences [5]

First, we learn the grammar that can represent the gap region in the structure. An example of the gap region is the bold symbols in the below sequences.

tactagcaatacgc ttgcgt tcggtggttaagtatg tataat gcgcgggcttgtcg

The grammar rule for the gap region part above will look like the following

X –(1.0)→ X B
X –(1.0)→ B B
B –(1.0)→ {a, t, c, g}

X represents the gap region. A gap is any sequence of symbols that their significant is not known yet. B represents the terminal symbols for all four bases.

After learning the gap region, the next region is the TTGACA region. The sub-sequence below in bold is the TTGACA region in the sequence.

tactagcaatacgc **ttgcgt** tcggtggttaagtatg tataat gcgcgggcttgtcgz

(%)	T	T	G	A	C	A
T	69	79	18	16	16	17
G	10	8	61	11	9	16
C	10	7	12	17	54	13
A	10	6	9	56	21	54

Probabilities for each symbol for the TTGACA region

(%)	T	A	T	A	A	T
T	77	12	60	12	15	82
G	8	6	14	14	8	5
C	10	6	11	13	20	7
A	5	76	15	61	56	6

Probabilities for each symbol for the TATAAT region

Length of Middle Space Region (bp)	Probability (%)
15	11.7
16	17.4
17	43.3
18	17.1
19	10.4

Probabilities for the length of the Middle Space region

Table 4.1: The Probabilities of Each Amino Acid for Each Region in an E. Coli Promoter DNA [40]

The TTGACA region's grammar is as follows

Y_1– (1.0)→X T_1	Y_2–(1.0) →Y_1 T_2	Y_3–(1.0)→Y_2 G_1	
Y_4– (1.0)→Y_3 A_1	Y_5– (1.0)→Y_4 C_1	Y_6–(1.0)→Y_5 A_2	
T_1–(0.69)→t	T_1–(0.10)→g	T_1–(0.10)→ c	T_1– (0.10)→a
T_2–(0.79)→t	T_2–(0.08)→g	T_2– (0.07)→c	T_2–(0.06) →a
G_1–(0.18)→t	G_1–(0.61)→g	G_1–(0.12) →c	G_1–(0.09)→a
A_1–(0.16)→t	A_1–(0.11)→g	A_1– (0.17)→c	A_1–(0.56)→a
C_1–(0.16)→t	C_1–(0.09)→g	C_1– (0.54)→c	C_1–(0.21)→a
A_2–(0.17)→t	A_2–(0.16)→g	A_2– (0.13)→ c	A_2–(0.54)→a

Y non-terminal represents the TTGACA region. T_1 is the location of the first symbol in the TTGACA region. T_2 represents the second symbol followed by G_1, A_1, C_1, and A_2.

For the middle region, it is represented using the non-terminal symbol M. A gap is any sequence of symbols that their significant is not known yet. B represents all four bases. The bold part in the below sequence is an example of the middle region of that sequence.

tactagcaatacgc ttgcgt **tcggtggttaagtatg** tataat gcgcgggcttgtcg

The middle region is parsed using the following grammar. The symbols for the middle region are represented using the letter M where M1 is the first symbol, M2 is the second, … etc.

M_2 –(1.0) →B B	M_3 –(1.0) →M_2 B	M_4 –(1.0) →M_3 B
M_5 –(1.0) →M_4 B	M_6 –(1.0) →M_5 B	M_7 –(1.0) →M_6 B
M_8 –(1.0) →M_7 B	M_{10}–(1.0) →M_9 B	M_{11}–(1.0) →M_{10} B
M_{12}–(1.0) →M_{11} B	M_{13} –(1.0) →M_{12} B	M_{14}–(1.0) →M_{13} B

M_{15}–(1.0) →M_{14} B M_{16} –(1.0) →M_{15} B M_{17}–(1.0) →M_{16} B
M_{18}–(1.0) →M_{17} B M_{19} –(1.0) →M_{18} B P_1 –(0.117)→Y_6 M_{15}
P_1 –(0.174)→Y_6 M_{16} P_1 –(0.433)→Y_6 M_{17} P_1 –(0.171)→ Y_6 M_{18}
P_1 –(0.104)→Y_6 M_{19}

Above, the Y non-terminal for the TTGACA region is concatenated with the various non- terminals of different lengths for the middle region M. The third region, TATAAT, follows the middle space region.

tactagcaatacgc ttgcgt tcggtggttaagtatg **tataat** gcgcgggcttgtcg

It is described using the following grammar:

Z_1– (1.0) →P_1 T_3 Z_2– (1.0) →Z_1 A_3 Z_3– (1.0) →Z_2 T_4
Z_4– (1.0) →Z_3 A_4 Z_5– (1.0) →Z_4 A_5 T_3–(0.77) →t
T_3–(0.08) →g T_3–(0.10)→c T_3–(0.05) →a
A_3–(0.12) → t T_4–(0.11)→c T_4–(0.15) →a
A_4–(0.12) →t A_4–(0.14)→g A_4–(0.13) → c
T_5–(0.82) →t T_5–(0.05)→g T_5–(0.07) →c
T_5–(0.06) →a Z_6– (1.0) →Z_5 T_5 A_3–(0.06) →g
A_4–(0.61) →a

Then finally the parser checks if the whole sequence can be parsed till the start symbol S. The grammar for S is

S –(1.0) → Z_6 B S –(1.0)→ S B

4.6 Results

A sample of 106 E Coli DNA promoter sequences is collected from the Machine Learning Repository UCI [39]. 53 positive and 53 negative sets were used as testing sets. E Coli promoter sequence was studied and analyzed.

The experiment was done on three stages. First, a context-free grammar was then built to describe the standard E Coli promoter's structure. Then, error correcting cover grammar was implemented over the previous grammar to measure distance from standard form. Finally, the experiment was repeated by modifying the parser and the grammar to incorporate probabilities of regions of the structure.

The experiment resulted in 5 false positives (%4.72) using the error count approach. This is equal to %95.283 accuracy. Accuracy was calculated using the following equation

$$\text{Accuracy} = \frac{True\ Positives + True\ Negatives}{Total\ Sample}$$

Errors count on rewrite rules was then replaced with probabilities. The parser was modified corresponding to the changes in the rewrite rules. Replacing error count with probabilities resulted in 4 false positives (%3.77) and accuracy of %96.226 (Figures 4.7 and 4.8).

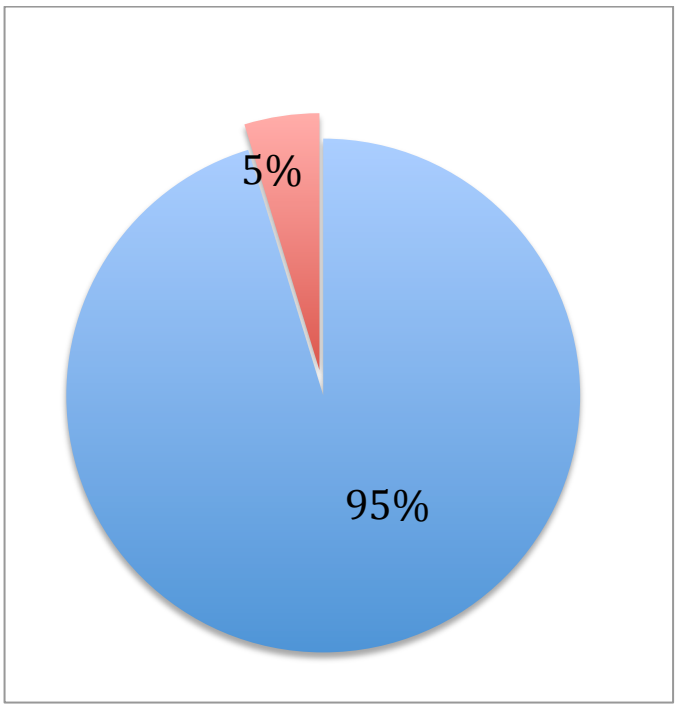


Figure 4.7: The percentage of false positive in red from total sample using an error correcting grammar

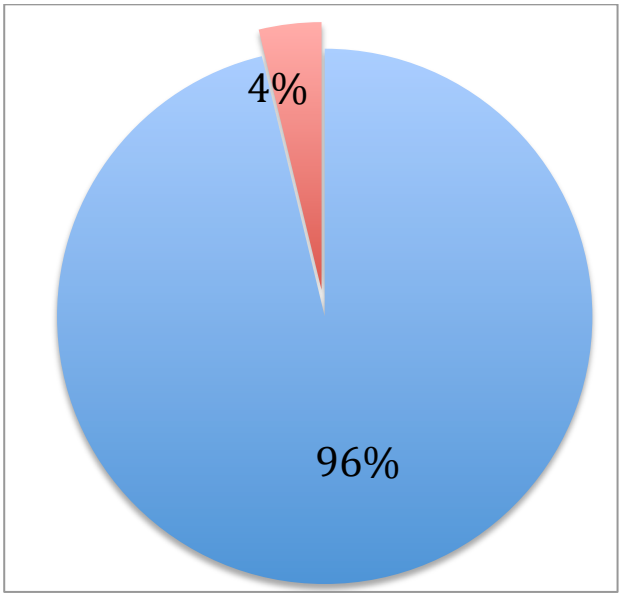


Figure 4.8: The percentage of false positive in red from total sample using probabilities

4.7 Contribution

Inferring a grammar and describing complex structures accurately, such as DNA, can be challenging. In this research, the general structure of the family class was inferred using an inference tool. However, that was not sufficient to describe the complex relationship between important motifs for the class adequately. Therefore, the grammar was inferred interactively. We have shown how the general description was modified using the class known standard structural characteristics. Then, an error correcting grammar was implemented on that definition. Later, that grammar was further improved by adding probabilities. The grammar was able to the class description showed to be more accurate and resulted in higher accuracy. Also, the approach is can be implemented using common programming languages an is not limited to a single one.

4.8 Conclusion

The aim of the chapter was to introduce an approach that is able to describe complex sequential structures. E. Coli Promoter DNA sequences were used to experiment the approach. The grammar was learned using interactive inference. The general grammatical structure was first inferred using Protomata-Learner. Then, it was modified based on the standard structure of E. Coli Promoter DNA sequences. The variation in the main motifs was incorporate in the grammar with the application of an error correcting grammar. Finally, probabilities were used in grammar for the next step instead of error count to improve results. Final results showed an

accuracy of %95.283 when applying the error correcting grammar to identify the E. Coli Promoter DNA sequences and %96.226 using probabilistic grammar.

Chapter 5

Data Mining and Classification Using Probabilistic Grammars

5.1 Introduction

So far, formal grammars were used to associate data sequences to a single family class. This research will focus on data mining heterogeneous sequences in large volumes that have various classes within by data mining using formal grammars (Figure 5.1). Data classification is defined as the process of arranging big data into smaller data categories to utilize this data more effectively and efficiently [51]. In this research, data family classes are identified and categorized based on their sequential structure. A well-implemented data classification benefits researchers in various ways. It helps retrieve and study a essential data type of interest effortlessly without the need to go through the large database [51]. It also provides exclusive structural and statistical information for each structural family pattern. This highlights the differences and similarities of data classes within the database. Another advantage data classification provides statistical information of the most frequent structure found in the data set. In addition, it assists effective and efficient sequence search by focusing searching for a specific sequence in a certain data class.

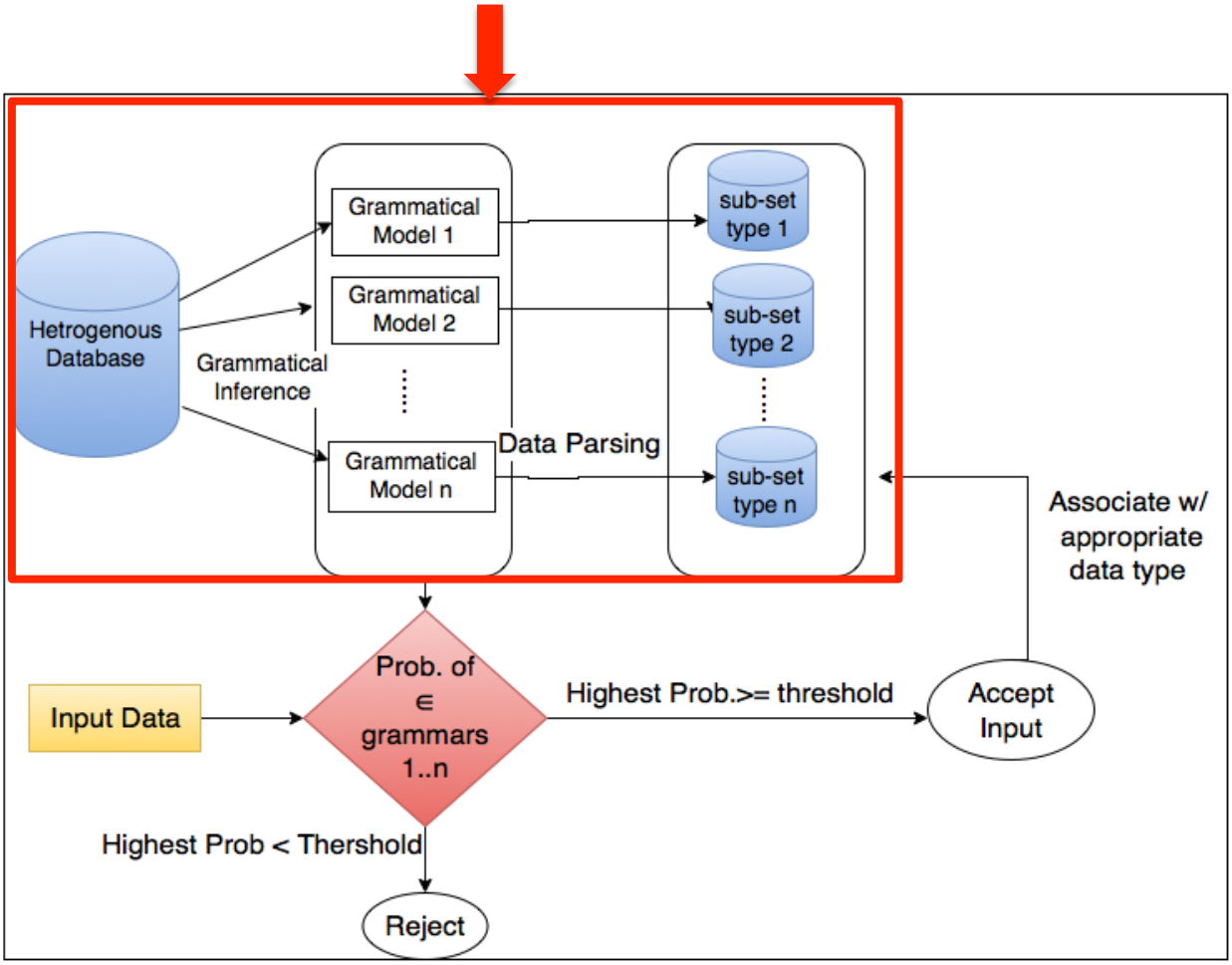


Figure 5.1: Data Classification Approach

This chapter is concerned in data mining and classification of big data. Compared to other data reduction models, such as k-means, this research uses a clustering approach based on the structure of data items. The technique utilizes the approach presented on previous chapters by applying it on various training sets of different classes. Inferring several grammars where each grammar represents a data class within the database.

In our experimentation, we used primate splice-junction DNA sequences collected from Fedorov Lab [64]. Splice-junction regions in DNAs are the ones that are concerned with RNA splicing. In some genes, the exons, which are the protein-coding regions, are interrupted by the introns, the non-coding regions. The introns are

removed in this process from the pre-mRNA and connects the exons together [58] .The location where the splice happens is the splice site. It is where the intron and exon are connected. The exon/intron site is called a donor site. Also, the intron/exon site is called an acceptor site. In the experimentation section of this research, that data will have a mix of sequences that either contains a donor site, an acceptor site, or neither. The model will analyze sequences and classify sequences that have the same splice-junction site together.

In this introduction, we have provided an overview and some important definitions. The next section will cover the research main challenges. In section 3, the section will describe briefly some other work done related to this research. Next, the work is explained in more details in section 4. Primate mammals splice-junction DNA sequences are used for experimentation in section 5. Results and discussion of the experimentation is explained in section 6. Finally, our research's contribution and final conclusion are in sections 7 and 8 consecutively.

5.2 Challenges

When classifying data there is an issue that needs to be considered. In sub-sets with similar structures, an input sequence can be accepted by more than one grammatical model and associated with more than one data set. In case the model associates the sequence to the first class that accepts it, it may not an accurate classification. Therefore, we had to find a method were it evaluate the data input and compare it to all candidate classes. Then, suing a probabilistic grammar, the data input is classified and

added to the family class with the highest probability. Data is not classified until it is run by all class parsers first and probabilities of being part f each class is generated. If it is accepted by more than one class, the data is associated with the class that has the highest probability.

Using the grammar, large data is split into smaller data sets that share a similar structure. When given a sequence, the input is parsed using different grammars. If more than one grammar accepts the sequence, these grammars are ranked in decreasing order according to the probability of the sequence they are parsed. It belongs to the first ranked grammar (Figure 5.2).

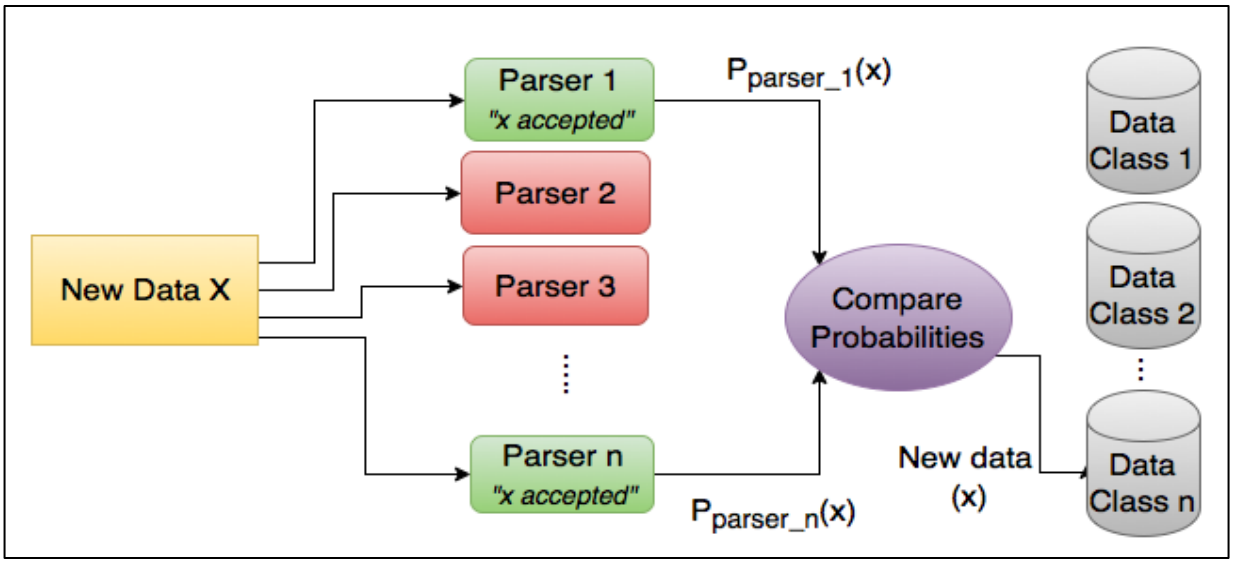


Figure 5.2: Associating a New Input to the Appropriate Class

5.3 Related work

Using formal grammars for data mining in general is not new. Borges and Levene

[12] have used hypertext probabilistic grammars to study user's website navigation. Based on a large data of previous user's navigation history, the algorithm learns the behavior, or the navigation pattern, of the user. Then, the algorithm is able to predict the next page the user is going to visit. Their algorithm is intended for real-time application whereas in this research we focus on static data.

Another paper written by Damasevicius used L-grammars to infer a description of splice junction sequences' structure [23]. The grammar is inferred using positive and negative examples. The model learnt was validates using a Support Vector Machine (SVM). Specifically, the outputs from the two models were compared for agreement. In this research we use a grammatical classifier for identifying splice junction sites.

5.4 Approach

This research aims to data mine big heterogeneous data. It learns the structural features, or grammatical rules, of each class within the large set. A structural model is then built using probabilistic context-free grammars. The grammar is inferred from training data with known classifications. The model is then used to classify unknown data points and associate them to the appropriate category of a family class based on it's structural characteristics. Classification is a well studied problem with many different approaches known for its solution [43]. Not many works have been done where formal grammars have been used for classification. Data mining using formal grammars are done three phases in our approach:

1. Inferring a grammar from the collected training data,

2. Building a data mining (grammar based) model, and
3. Using the grammatical models.

These phases are described in detail next. Biological DNA sequences of splice-junction regions of primate mammals are used for experimentation in this research.

A. Inferring a Grammar

In this step our objective is to extract the statistical and structural features of data and encode them as grammars. In order to do this, several steps are required. First, sufficient training sample data class is collected that can represent the intended class. For each single class within the large data set, a sufficient sample is collected to represent the family class. Then, a grammar is inferred for each class using the appropriate training sample. Also, probabilities of the grammar's production rules are calculated in this step. In this research, the grammar was learnt using an interactive inference method. The general grammatical description is learnt using a grammatical tool, Protomata-Leaner [20][19]. Then, It is modified using a teacher [9].

B. Building a Data Mining Model

Using the inferred grammar for the class, described in the previous step, a grammatical model is built. The grammatical model can be implemented using any common programming language. In this research, C++ was used for coding the grammatical model. The model receives a DNA sequence as an input. Then, the model parses the input and generates the probability of the sequence being part of that class.

If the probability is less than a certain pre-set threshold, the input is rejected as being part of that class. This is repeated using all models. After the sequence is processed by all models, the probability of that input sequence being part of each of those classes is generated. Figure 5.3 illustrates a grammatical model for a single class.

C. Using Data Mining Models

Given large data containing heterogeneous data types, the grammatical model developed in the previous step is used to build a tool to classify big data. This is repeated using all models. We build multiple grammatical models, one for each class. These grammatical models will be applied on unknown data. Any data point will go through all the grammatical models. Each grammatical model will parse the sequence and generate the probability of the point belonging to that data class. When a data point has gone through all the data models, there will be probabilities of this point belonging to the different classes. It results the probability of that input sequence being part of each of those classes. If the probability of the sequences belonging to a certain class is above a pre-set threshold, it is accepted as part of that class. If the input is accepted by more than one class, as a simple resolution mechanism, the data point can be then associated with the class with the highest probability. If the data probability is less than a pre-set threshold for all the classes, the data point is then identified as not part of any class. When this process is complete, all the points in the big data will be partitioned into clusters, where each cluster has points with similar structural features. The overall process is illustrated in figure 5.2.

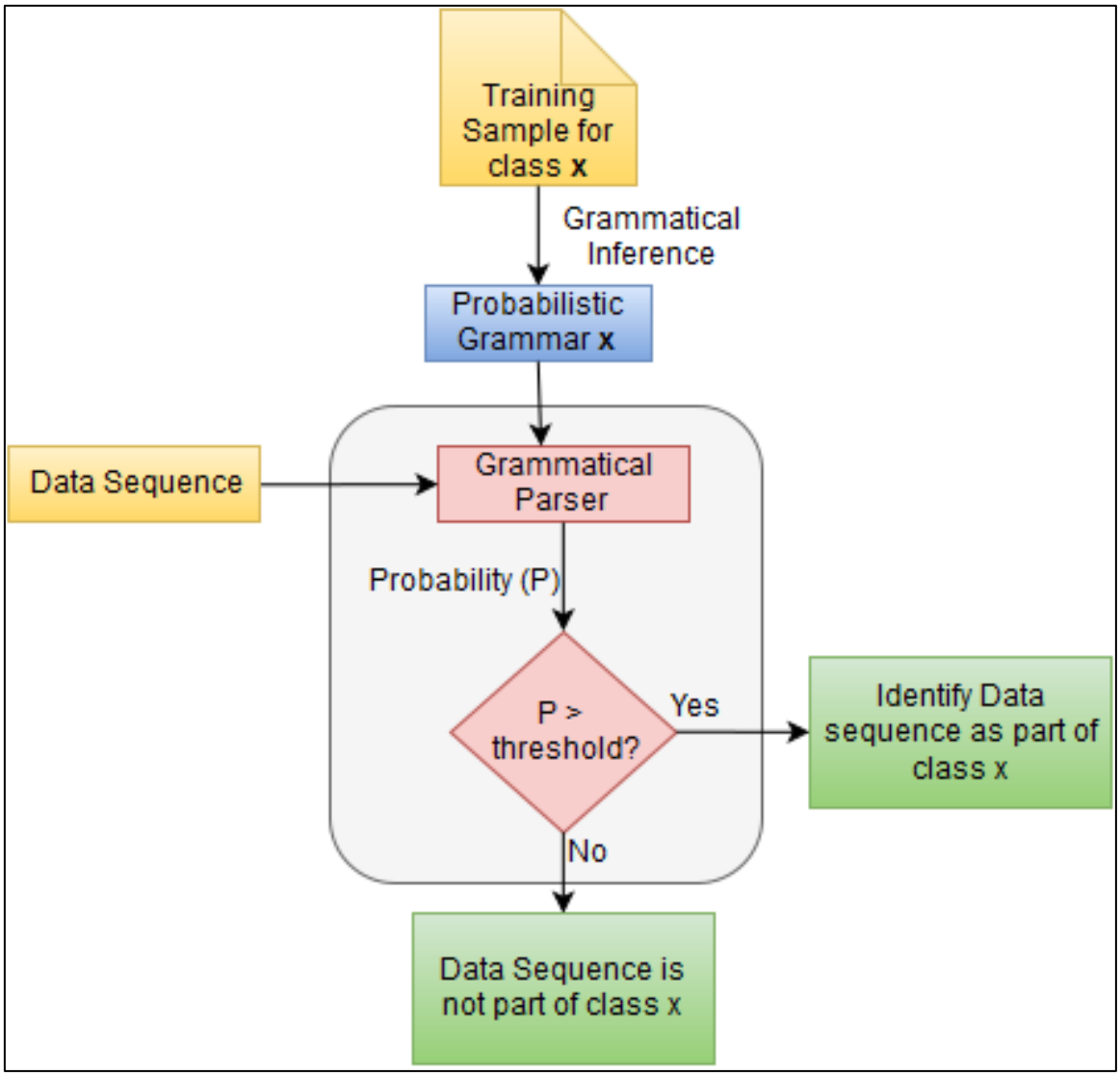


Figure 5.3: Building a Grammatical Model for Some Class x [3]

In some cases, classes might need to be partitioned into smaller sub-classes. A grammar is inferred for each sub-class. When the sub-classes need to be joined to get the overall classes grammar, they can be joined using the union operation U (Figure 5.4) where $G_{class} = G_{sub1} + G_{sub2} + \ldots + G_{sub\ n}$

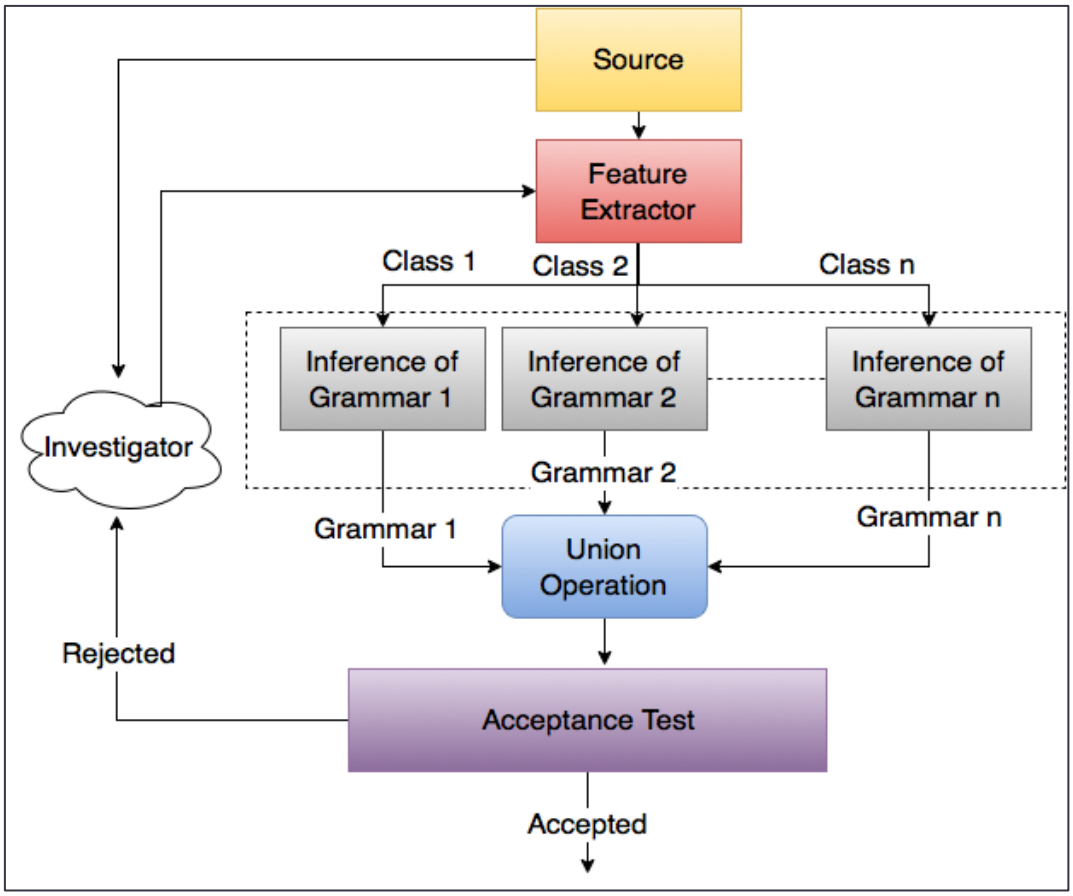


Figure 5.4: Partitioning Model [5]

Moreover, as more new data is introduced to the class, the accuracy of the model might change. As more data with different structures are presented, we might find the current grammatical model is over-fitting with high false negatives, or over-generalized with high false positives. To overcome this issue, a model adjustment is required. A new grammar is inferred while including the new data collected in the training sample. The new grammar then replaces the newer one to improve accuracy (Figure 5.5)

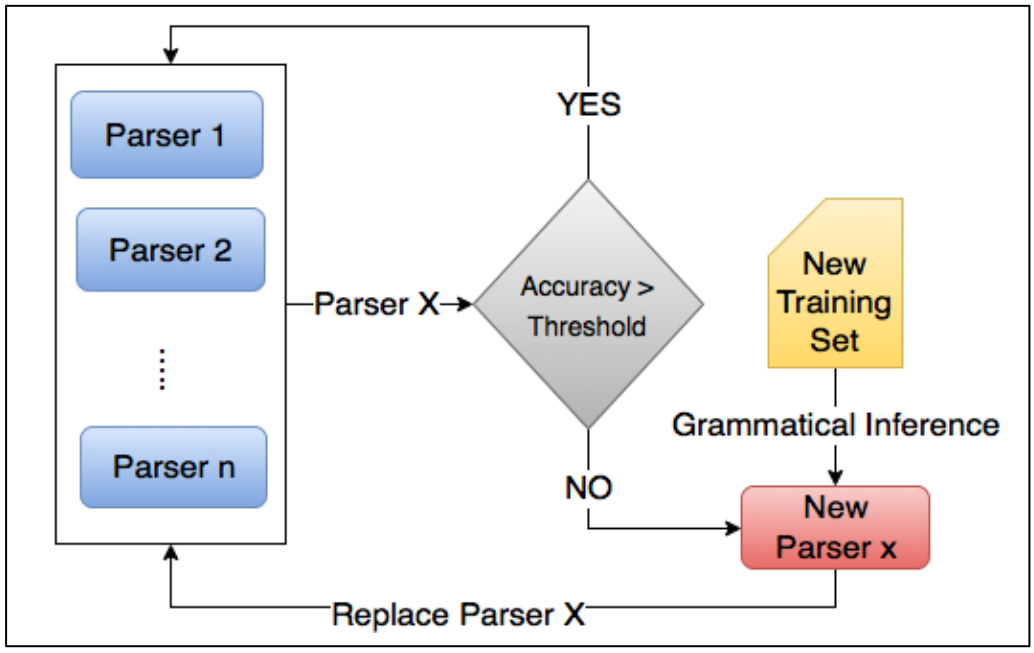


Figure 5.5: Model Adjustment

5.5 Experimentation

The goal of the experiment is to successfully classify DNA sequences into correct classes. Here, there are three classes: donor, acceptor, or neither. An intron is the nucleotide sequence region in a DNA that is removed in the RNA splicing process. Sequences in DNAs that are joined together after the splicing process are exons [58]. DNA sequences are classified into three categorized exon/intron (EI) site (also called donor), intron/exon (IE) site (also called acceptor), or neither (N). In this paper we address the classification of a DNA sequence into the above three categories as a case study. Experimentation data have been collected from Fedorov Lab [64]. A total of

586,833 sequences were collected for experimentation of which 172,087 sequences are donor site, 220,470 sequences are acceptor site, and 194,276 sequences are negative sample which are neither donors nor acceptors. The length of each sequence read was 60 bp.

A. Inferring a Probabilistic Context-Free Grammar

To infer a grammar, we first collect a random sample for training to represent the data class. A total of 200 sequences were randomly collected from the acceptor and donor data sets. Each group of training sample were fed separately into Protomata-Learner. Protomata-Learner utilizes regular grammar to extract the structural description [20][19]. The structural graphical representation can for an acceptor splice-junction site can be seen in figure 5.6 below.

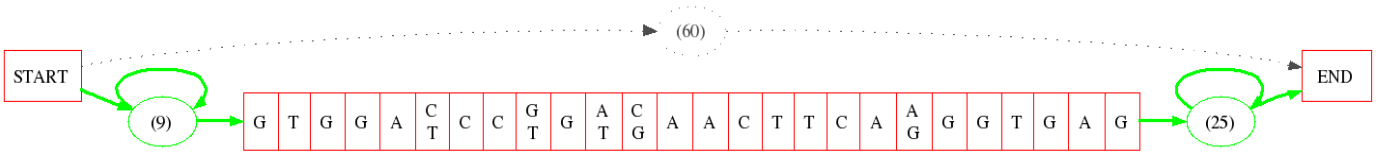


Figure 5.6: A Structral Description of an Acceptor Splice-Junction Site as Generated by Protomata-Learner

To improve the results we included an interactive inference. To infer a grammar for the splice site sequences, significant motifs are studied first. Almost all donor sites contain GT and acceptor sites contain AG. They are the most distinctive motif for donor and acceptor splice sites identification. Although they are significant, they are insufficient for correct identification [57]. In order to improve accuracy we use

probabilistic grammar to calculate the probabilities of the symbols around the GT and AG motif. This gives a better insight of the sequence structure and provides better accuracy. Using the collected sample, the probability of each of the four symbols a, t, c, and g being in each location in the sequence is calculated. If the standard variation is high in a certain location, this location is considered significant. It is then used as part of the structural description for that class. If the variation is not significant or probabilities are almost equal for all the four symbols (close to %25 for each symbol), it is considered as a gap symbol and insignificant for this research. Figure 5.6 shows an example of a parsing tree. In figure 5.7 below, the X0 symbols are the locations that were labeled as gaps based on their probability. T1, …, T12 are locations where symbols had probability variation thus were identified as significant. This will reduce the number of rules where only rules with significant structural importance are examined.

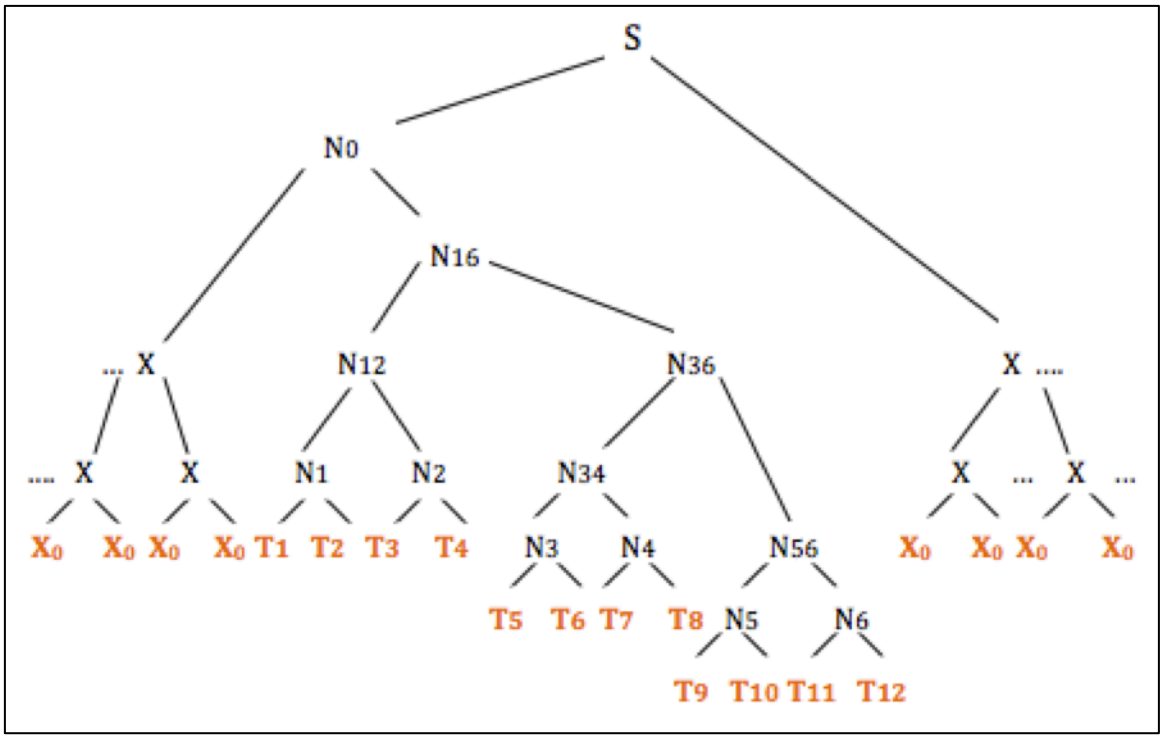


Figure 5.7: An example Parsing Tree for Donor Site Sequence where X0 is a Gap and Tx is a Significant Symbol for Structural Identification [3]

Mount published a paper in biomedical engineering that has already calculated the frequency of the four symbols in exon and intron sequences [44]. The symbol's probability in each base pair (bp) location is calculated using a sufficient training sample. These probabilities are integrated into the grammar to produce a probabilistic context-free grammar for both donor and acceptor sites. Using these probabilities, they can provide a better accuracy to the grammatical model. The probabilistic context-free grammar for donor splice-junction site can be found in figure 5.8 .

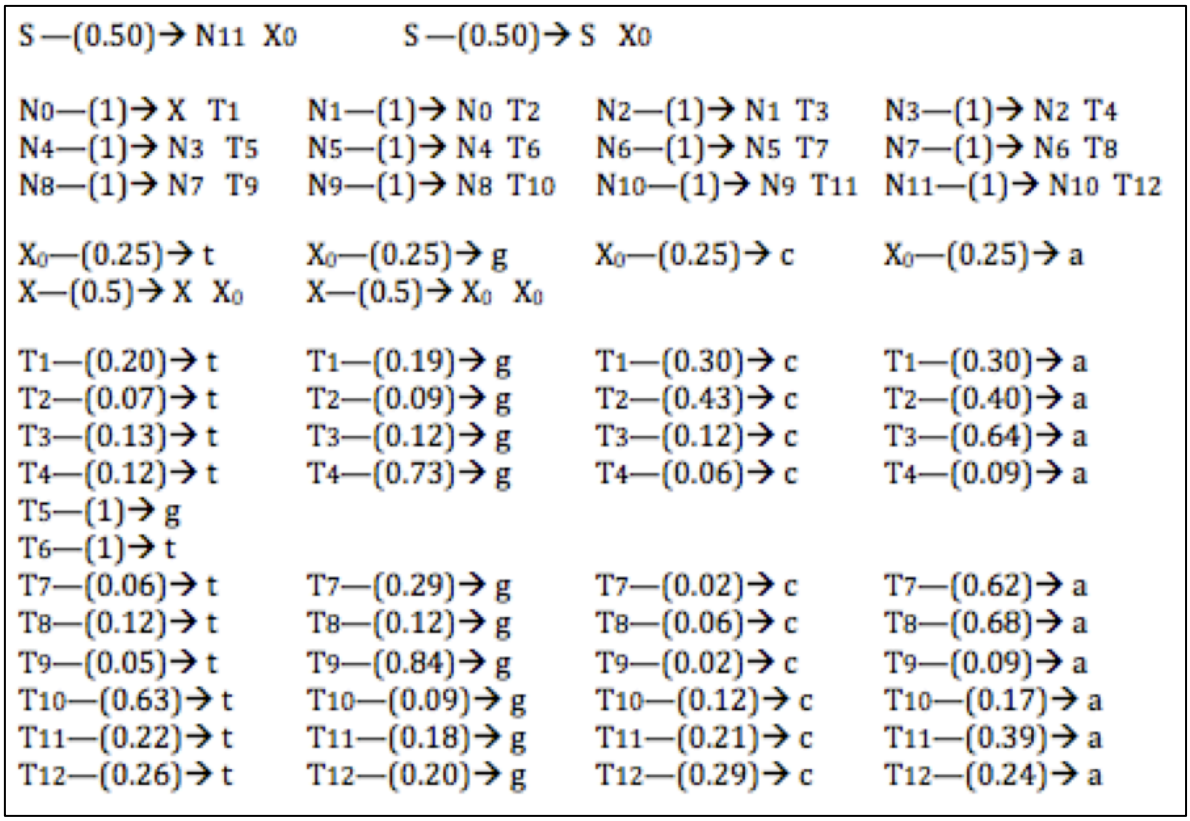


Figure 5.8: An example donor probabilistic context-free grammar [3]

In the grammatical rules, the X0, which resembles, has an equal probability of happening for all four nucleic acids. The probabilities varies for a region of 12 nucleic acids sequences symbolized with T1- T12. We can see that the “GT” part has the probability of %100 in donor splice sites.

B. Building a Grammatical Module

After completing the grammatical inference for all classes within the big data, we need to use the grammars. In this phase, each grammar is encoded into a parser that represents the class. Those parsers can be can be built using any common

programming language. In our research we have used C++. These models accept a sequence and runs it through each parser. Each parser will calculate the probability of the sequence being part of its class. If the probability is less or equal to a certain pre-set threshold, the parser will reject it. If it is higher, it will accept it as part of the class.

C. Classification

When there is heterogeneous big data, various grammatical parsers are used to classify data inputs. In some occasions, these data are accepted by more than one parser. The model has to select the correct class to associate that data input with. In our example, we have two class, i.e., donors and acceptors. The program reads DNA sequences from any heterogeneous big data set and runs them through each grammatical module. If more than one module accepts a sequence as a part of their class, the program will associate the sequence with the class with the highest probability. Then the sequence is added to the class set it is associated with as can be seen in figure 5.9. If it is rejected by both the modules, it will be added to the neither set.

D. Introducing New Data

When new data is introduced to the big data, it goes through a similar process. The new data will run through the grammatical model. Each data point gets parsed by all class parsers. When probabilities are generated for all classes, the new data point is associated with the class that has the highest probability.

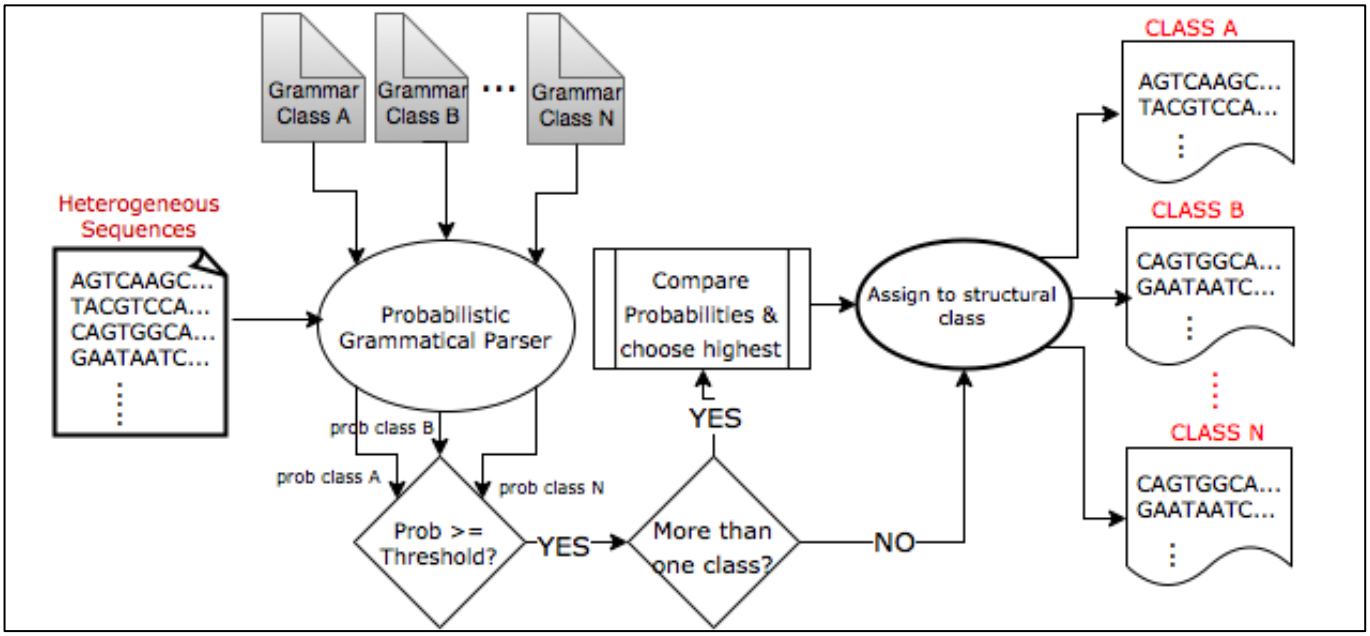


Figure 5.9: Classifying Heterogeneous Data Using Grammatical Modules

5.6 Results and Discussion

A total of 586,833 sequences containing 172,087 donor sites, 220,470 acceptor sites, and 194,276 are neither a donor or an acceptor (negative sample). The length of each sequence read was 60 pb long. The splice-junction DNA sequences were fed to the grammatical data mining model. After all data were processed, the experimentation generated the following results:

- We calculated the total accuracy and sensitivity using the following equations:

$$\text{Accuracy} = \frac{True\ Positives + True\ Negatives}{Total\ Sample}$$

$$\text{Sensitivity (Recall)} = \frac{True\ Positives}{Total\ Sample}$$

It results in total of %73.65 accuracy as can be seen in Table 5.1.

	Accuracy	Sensitivity (Recall)
Acceptor	%73.63	%76.04
Donor	%72.21	%73.69
Total	**%73.65**	**%75.01**

Table 5.1: Experimentation Accuracy and Recall

In our research, we have introduced a way that formal grammars can be used to data mine big data. Splice-junction DNA sequences were chosen to experiment our approach. Various papers have been published to identify the splice-junction sites in DNAs with higher accuracy results (see e.g., [59][61]). However, the main focus of this paper is not on identifying the splice-junction sites in DNAs. Rather, our intention is to demonstrate the applicability of PCFGs in solving different problems. We focus on data mining using structural sequences with different classes in general using PCFGs.

5.7 Contribution

There are various approaches that are available in literature to data mine big data. In our research, we provided an approach that utilizes probabilistic grammars for data mining. Formal grammars are very valuable in describing structural data. They have the advantage of describing complex structures. They also able to overcome location base structural characteristics. In addition, providing statistical data through

probabilistic grammars. Here, we proposed a way to use formal grammars and experimented on biological data. As future work, this can be improved by combining it with other algorithms and approaches for more efficient data mining using grammars.

5.8 Conclusion

This paper presents an approach to data mine using probabilistic context free grammars. The idea is to identify data as part of a class based on its structural characteristics. It generates a probability of a data being part of various data classes. The approach has a lot of potential since it can be applied to various types of structural data. The approach will provide a structural insight and statistical information of different data classes within big data. One way this approach can be further improved is by experimenting integrating this approach with other methods. However, it is taken in consideration as future work.

Chapter 6

Conclusion and Future Work

This chapter provides a summary of the overall research work done in this dissertation. It provides an overview of data mining big data using grammars. The process was introduced in previous chapter in various phases: inferring a structural model for a single class using formal grammar, associating complex sequences to a data class using these models, and data mining heterogenous big data through classifying them into a number of classes using formal models. After the summary in next section, contributions and future work are then listed in section 6.2.

6.1 Summary

In this dissertation, the main motivation is explore new ways in data mining heterogenous big data. We have utilized formal grammars to achieve this purpose. The main objectives of this dissertation are:

1) **Learning Structural Model**

   Structural models are important for describing the pattern for a data class. In this research, we have utilized formal grammars to develop structural models to represent a class. These models are developed as the first step in the process of data mining using grammars.

2) **Data association**

   Another objective is associating a given data point to a class successfully by identifying the data's structure and comparing it to the grammatical description of

a class. After the model generates the probability of the input being part of the class, the data point is then either accepted or rejected.

3) **Data classification**

When given a large data set with different classes, the algorithm is able to classify different data points to the appropriate data class. After all data inputs are processed, the big data is partitioned into smaller categorizes that share similar structure.

4) **Building a composite model for big data complex sequences**

Some structures have inter-relationships parts of their structure. It is critical for the class identification to recognize such correlation within the data pattern. We have resolved this issue through interactive inference and utilizing grammatical tools. Interactive inference further improves the inferred grammar to avoid preventable over-generalization or over-fitting. Also, to simplify a complex structure, a grammars for each of the segments within the structure is inferred individually. Then, concatenation was used to link these segments together. This also helps highlight correlation between parts of the structure.

These objectives are met by doing it in several steps. Each step is explained in a dedicated chapter. The steps are:

a) Building a structural model using formal grammars for a single class. A grammar is inferred to describe the structure of a class. A structural model that recognizes and compares inputs to the structure of the class is built. It does that by parsing

inputs and analyzing structure to the grammatical description of the class. We used protein sequences as a case study for this approach.

b) The second chapter of this dissertation explained an approach for more complex sequences. We have used interactive grammatical inference. The general structure is first inferred using a tool. Then, a teacher further improves the grammar. The grammar is split into parts based on the different significant parts within the sequences. The model parses each part individually. Then, when the part is successfully parsed and accepted, it is concatenated with the other parsed parts. When it is done, it will generate the overall structure. If one part cannot be parsed, the whole sequence is rejected as part of the class. E. Coli DNA promoter sequences were used for this approach as a case study. We used probabilistic context-free grammar to build the structural model.

c) The this phase of the dissertation includes using structural model to data mine heterogeneous big data that includes various classes. A probabilistic grammar is inferred for each class within the big data. Using these grammars, we built a structural model for each class appropriately. Then, all data points in the big data are processed and parsed by the probabilitic grammar. The probability of a data point being part of each of the classes is generated. If that probability is less than a pre-set threshold, the input is rejected as part of the class. Otherwise, the input is accepted. If the input is accepted to more than one class, the input is associated with the class with the higher probability. In figure 6.1, we can see an

illustration of the overall process. Splice-Junction DNA sequences were used for experimentation.

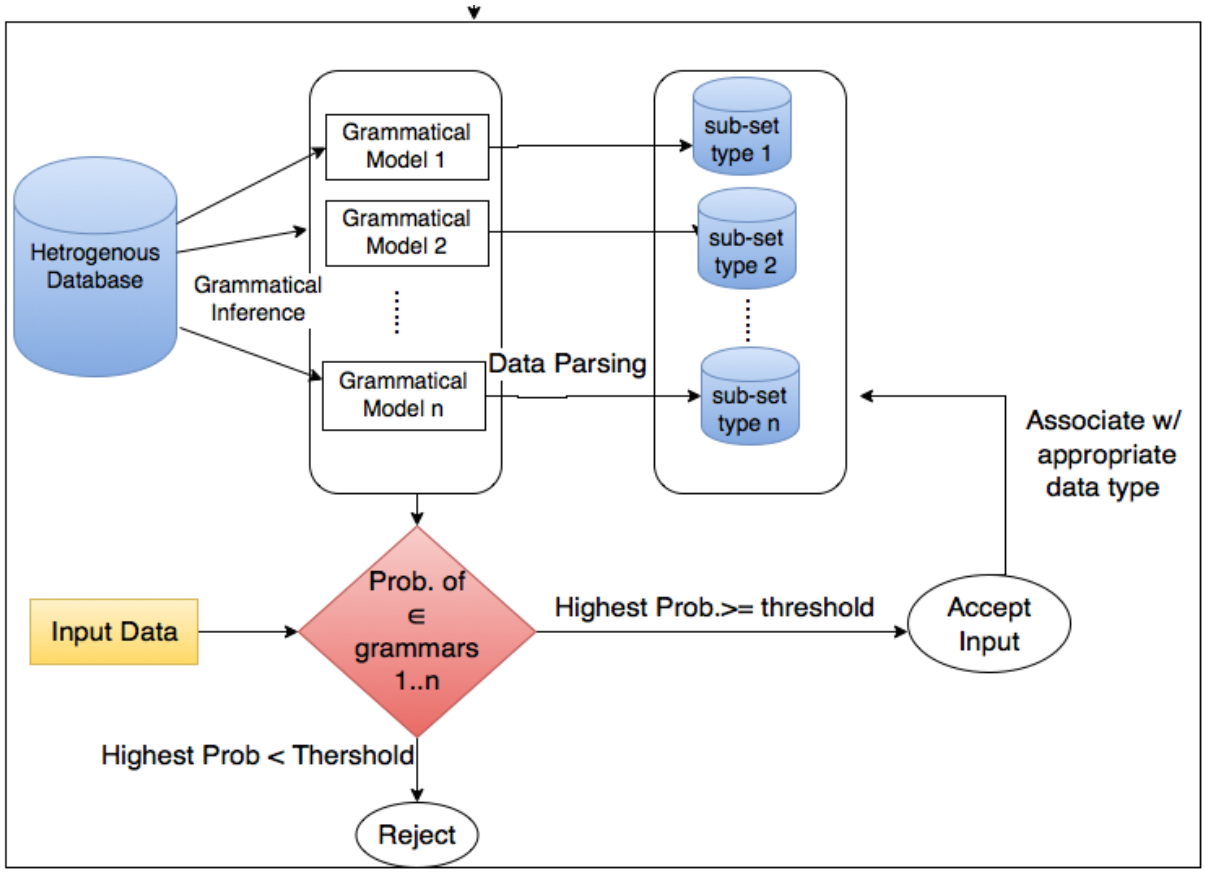


Figure 6.1.: Overall Process For Data Mining Heterogeneous Complex Data Using Grammars

6.2 Contributions

A new method offered for data mining that utilizes formal grammars to extract information and describes large data in a compact matter.

Three approaches are proposed which include:

1. **Structural model for data sequences using formal grammars**

   We have presented an approach were a structural model is built using formal grammars. Then a case study was given using protein sequences were the model identifies sequences that belong to the class based on their structural characteristics.

2. **Structural model for complex sequences**

   We have explain an approach that can used to build a structural model for complex sequences using formal grammars. Using concatenation as a tool and inferring probabilistic grammar using interactive inference, we were able to achieve that objective.

3. **Building a composite model to data mine heterogeneous big data**

   Formal grammar is a valuable and advanced tool for data association, extraction, and modeling. We have shown an method to apply grammars for data mining big data with different structural classes. Using grammars, we were able to classify the big data into smaller classes that share similar structural characteristics.

6.3 Conclusion and Future Work

6.3.1 Conclusion

Formal grammars have effectively been used in a various areas such as natural language processing [42], bioinformatics [53], and applied behavior analysis [6]. They

proved to be capable in describing the syntax of a language or the structural relations in patterns or data [26]. However, not many have used formal grammars in data mining.

In our work, we have shown how grammars are an effective and valuable tool that can be benefited in data mining. We have successfully applied formal grammars for data mining big data. The advantage of using grammars to data mine is that, in addition to data mining, they provide structural description in a condense matter. Probabilistic grammar are also able to provide statistical information about the data. It also able to overcome location based identification of structural regions.

6.3.2 Future Work

The work found in this research can be further extended in different ways. Below is a list of some future work that can be done:

- **Automation:** One way we can further extended our work in data mining using grammars is by providing an automated way to for maintaining the models accuracy. The process is discussed in chapter 5, however, the new grammar has to be done interactively with an interference of a teacher. We aim on doing the maintenance so that when the parsing error reaches a certain the threshold, it will generate a process where it modifies the grammar foe a better accuracy.
- **Multi-Dimensional Data:** Also, another direction that we can work on to extend this research is by applying it on multidimensional big data. This is done by inferring a grammar to represent each dimension. Then apply integration to obtain the overall data by including multi-dimensional grammars.

- **Other Applications:** We would like to extend our work through applying it on other applications such as scaffold based drug discovery and predicting interaction sites in proteins.

Bibliography

[1] A. Algwaiz, “Formal Modeling and Mining of Big Data“, Dissertation Proposal, March 2016.

[2] A. Algwaiz, R. Ammar, and S. Rajasekaran. "Framework for Data Mining of Big Data Using Probabilistic Grammars". In *e-Learning (econf), 2015 Fifth International Conference on*. IEEE, 2015.

[3] A. Algwaiz, S. Rajasekaran, and R. Ammar. “Data Mining Using Probabilistic Grammars”. In *the 16th IEEE International Symposium on Signal Processing and Information Technology (ISSPIT 2016)*. IEEE International Symposium on IEEE, 2016.

[4] A. Algwaiz, S. Rajasekaran, and R. Ammar. "Predicting E. Coli promoters using formal grammars", In *Signal Processing and Information Technology (ISSPIT*), 2015 IEEE International Symposium on IEEE, 2015.

[5] R. Ammar. “Interactive Grammatical Inference”, M.S., Electrical and Computer Engineering Department, University of Connecticut, May 1981.

[6] R. Ammar. “Software Optimization Using User Models”, Ph. D., Electrical and Computer Engineering Department, University of Connecticut, December 1983.

[7] R. Ammar, S. Demurjian, J. Greenshields, K. Pattipati, and S. Rajasekaran, "Analysis of Heterogeneous Data in Ultrahigh Dimensions." In *Emergent Information Technologies and Enabling Policies for Counter Terrorism*. R. Popp and j. Yen (eds.), IEEE Press, (2006).

[8] S. S. Anand, A. R. Patrick, J. G. Hughes, and D. A. Bell. "A Data Mining Methodology for Cross-Sales". Knowledge-based Systems, (1998) ,pp. 449 – 461.

[9] D. Angluin. "Learning Regular Sets from Queries and Counterexamples", Inform. and Comput., 75 (1987), pp. 87–106

[10] I. Ben-Gal. "Bayesian Networks", in Encyclopedia of Statistics in Quality & Reliability, Wiley & Sons (2007).

[11] HKD. Hn. Bhadeshia. "Neural networks in materials science." *ISIJ international* 39.10 (1999): 966-979.

[12] J. Borges, and M. Levene. "Data mining of user navigation patterns." *Web usage analysis and user profiling*. Springer Berlin Heidelberg, 2000. 92-112.

[13] C. Bystroff, and A. Krogh. "Hidden Markov Models for Prediction of Protein Features", Protein Structure Prediction (2008): 173-198.

[14] P. Cabena, P. Hadjinian, R. Stadler, J. Verhees, and A. Zanasi, "Discovering Data Mining —From Concept to Implementation", Englewood Cliffs, NJ: Prentice-Hall, 1998.

[15] A. Califano. "SPLASH: Structural Pattern Localization Analysis by Sequential Histograms", *Bioinformatics* 16.4 (2000): 341-357.

[16] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth. "CRISP-DM 1.0 Step-by-step data mining guide." (2000).

[17] N. Chomsky, “Three Models for the Description of Language”. In *IRE Transactions on Information Theory*. September, 1956.

[18] K. J. Cios, W. Pedrycz, and R. W. Swiniarski. Data Mining Methods for Knowledge Discovery. Vol. 458. Springer Science & Business Media, 2012: 1-26.

[19] F. Coste, and G. Kerbellec. “A Similar Fragments Merging Approach to Learn Automata on Proteins”. In *ECML,* 2005.

[20] F. Coste, F. Kerbellec. “Learning Automata on Protein Sequences”. In *JOBIM*, Jul 2006, Bordeaux, France: 199-210.

[21] F. Coste. “Tutorial on Modelling Biological Sequences by Grammatical Inference: Bibliography”. In *Symbiose Project IRISA/INRIA Rennes-Bretagne Atlantique*, France, (2010): 107-151

[22] W. Daelemans. "Colin de la Higuera: Grammatical inference: learning automata and grammars". Machine Translation, vol. 24, no. 3-4, (2010): 291-293.

[23] R. Damasevicius. “Structural Analysis of Regulatory DNA Sequences Using Grammar Inference and Support Vector Machine”. In *Neurocomputing* 73, no. 4, (2010): 633-638.

[24] R. Durbin, S. Eddy, A. Krogh, and G. Mitch.” Biological sequence analysis: Probabilistic Models of proteins and nucleic acid” .University press, Cambridge, 1998

[25] U. Fayyad, R. Uthurusamy, “Evolving Data Into Mining Solutions For Insights”. In *Communications of the ACM*. 2002 Aug 1;45(8):28-31.

[26] K.S. Fu, and T. L. Booth. "Grammatical inference: Introduction and survey-Part I." In *IEEE Transactions on Systems, Man, and Cybernetics* 1.SMC-5 (1975): 95-111.

[27] H. A. Gabbar, “Modern formal methods and applications”, Springer Science & Business Media, 2006

[28] Y. Gahi, M. Guennoun, H.T. Mouftah, “Big Data Analytics: Security and privacy challenges”, In *Computers and Communication (ISCC), IEEE Symposium*, 2016 Jun: 952-957.

[29] A. Gandomi, and M. Haider, “Beyond the hype: Big data concepts, methods, and analytics”, International Journal of Information Management, Volume 35, Issue 2, 2015 April: 137–144.

[30] J. Gantz and D. Reinsel, *“The Digital Universe in 2020: Big Data Bigger Digital Shadows Biggest Growth in the Far East”,* In *IDC iView: IDC Analyze the future*. 2007, 2012 Dec:1-6.

[31] E.M. Gold. “Language Identification in the Limit”, Information and Control, 10,1967:447– 474 .

[32] F.V. Jensen. *An introduction to Bayesian networks*. Vol. 210. London: UCL press, 1996.

[33] D. Klein, and C.D. Manning. "Fast Exact Inference with a Factored Model for Natural Language Parsing". *In Advances in Neural Information Processing Systems 15 (NIPS 2002)*, 2003, Cambridge, MA: MIT Press, pp. 3-10.

[34] I. Jonassen, J.F. Collins, and D.G. Higgins. "Finding flexible patterns in unaligned protein sequences."In *Protein science* 4.8 (1995): 1587-1595.

[35] D. Jurafsky, and J.H. Martin. "Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition", Prentice Hall, p. 469

[36] R. *Kohavi.* "A study of cross-validation and bootstrap for accuracy estimation and model selection".In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI), 1995.*

[37] A. Krogh, I.S. Mian, and D. Haussler. "A hidden Markov model that finds genes in E.coli DNA", In *Nucleic Acids Research*, 1994, Vol. 22, No. 22, Sept 1994.

[38] S. Leung, C. Mellish, and D. Robertson. "Basic gene grammars and DNA-ChartParser for language processing of Escherichia coli promoter DNA sequences".In *Bioinformatics* 17, 2010: 226–23.

[39] M. Lichman. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, (2013).

[40] S. Lisser and H. Margalit, "Compilation of E. Coli mRNA Promoter Sequences", Nucl Acids Res, vol. 21, no. 7, pp. 1507X1516, 1993.

[41] O.L. Mangasarian. "Data mining via support vector machines." *System Modeling and Optimization XX*. Springer US, 2003: 91-112.

[42] C.D. Manning, and H. Schütze. *Foundations of statistical natural language processing*. Vol. 999. Cambridge: MIT press, 1999.

[43] D. Michie, D.J. Spiegelhalter, and C.C. Taylor. "Machine learning, neural and statistical classification." (1994).

[44] S.M. Mount. "A catalogue of splice junction sequences." *Nucleic acids research* 10.2 (1982): 459-472.

[45] R. Narayanan, B. Ozisikyilmaz, J. Zambreno, G. Memik, A. Choudhary. “Minebench: A Benchmark Suite For Data Mining Workloads”. In *2006 IEEE International Symposium on Workload Characterization,* 2006 Oct 25 :182-188.

[46] C.G. Nevill-Manning, T.D. Wu, and D.L. Brutlag. "Highly specific protein sequence motifs for genome analysis". In *Proceedings of the National Academy of Sciences* 95.11 (1998): 5865-5871.

[47] C.G. Nevill-Manning, and I.H. Witten. “Identifying Hierarchical Structure in Sequences:A linear-time algorithm“, Journal of Artificial Intelligence Research 7 (1997) 67–82.

[48] M. Pagni, L. Cerutti, and L. Bordoli. “An introduction to Patterns, Profiles, HMMs and PSI-PRED”, (2003)

[49] G. Piatetsky-Shapiro. *Advances in knowledge discovery and data mining*. Eds. Usama M. Fayyad, Padhraic Smyth, and Ramasamy Uthurusamy. Vol. 21. Menlo Park: AAAI press, 1996.

[50] S. Rajasekaran, and M. Nicolae. "An error correcting parser for context-free grammars that takes less than cubic time". In *International Conference on Language and Automata Theory and Applications*. Springer International Publishing, 2016.

[51] V. Ramakrishnan, P. Venugopal, T. Mukherjee. “Proceedings of the International Conference on Information Engineering, Management and Security 2015”, In *Association of Scientists, Developers and Faculties (ASDF)*, 2015.

[52] I. Rigoutsos, and A. Floratos. "Combinatorial pattern discovery in biological sequences: The TEIRESIAS algorithm." *Bioinformatics* 14.1 (1998): 55-67.

[53] D.B. Searls. "The linguistics of DNA." *American Scientist* 80.6 (1992): 579-591.

[54] P. Senin, J. Lin, X. Wang, T. Oates, S. Gandh, A.P. Boedihardjo, C. Chen, S. Frankenstein, and M. Lerner, “Grammarviz 2.0: A Tool For Grammar-Based Pattern Discovery In Time Series”**,** In *Ecml/Pkdd Conference*, 2014

[55] T. Srivastava. “ Introduction to k-neartest neighbors: Simpilified”, Analytics Vidhya, Oct 2014.

[56] T.A. Sudkamp. “Languages and Machines: An Introduction to the Theory of Computer Science”, Addison-Wesley Longman Publishing Co., Inc., MA, USA, 1988, Pages 30-32.

[57] T. Strachan, A. Read, “Molecular Genetics”, third ed. Garland Publishing Inc, 2003.

[58] “Transcription and Translation: RNA Splicing”, DNA Learning Center, Cold Spring

Harbor Laboratory, Internet: https://www.dnalc.org/resources/3d/rna-splicing.html , [03/01/2017]

[59] C. Trapnell, L. Pachter, and S. L. Salzberg."TopHat: discovering splice junctions with RNA-seq*". In Bioinformatics 25*, 2009: 1105-1111.

[60] UniProt Consortium, "UniProt: a hub for protein information", Nucleic Acids Res. 43: D204-D212 (2015), http://www.uniprot.org

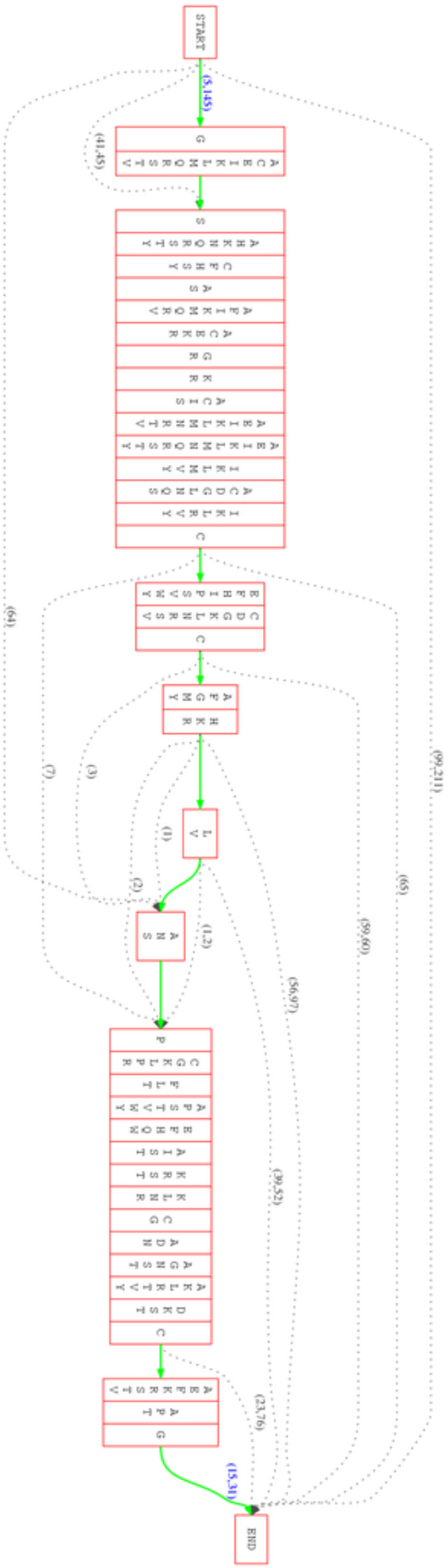
[61] K. Wang, D. Singh, Z. Zeng, S. J. Coleman, Y. Huang, G. L. Savich,X. He, Piotr Mieczkowski, S. A. Grimm, C. M. Perou, J. N. MacLeod, D. Y. Chiang, J. F. Prins, and J. Liu, "MapSplice: Accurate mapping of RNA-seq reads for splice junction discovery". Nucleic Acids Res. 2010 Oct; 38(18): e178, Aug 2010.

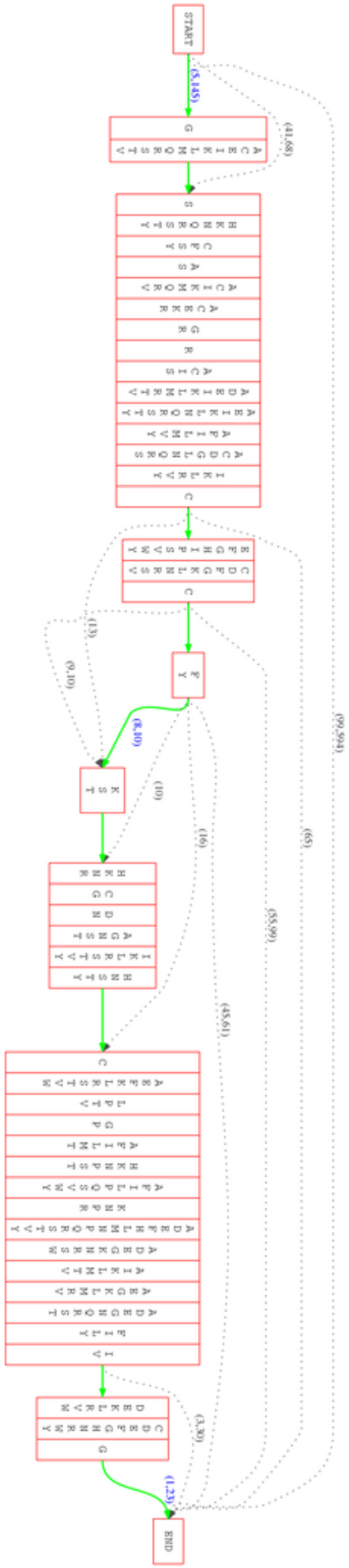
[62] X. Xiao, P. Wang, and K. Chou, “Predicting protein structural classes with pseudo amino acid composition: An approach using geometric moments of cellular automaton image“, Journal of Theoretical Biology , Volume 254, Issue 3, 7 October 2008, Pages 691–696

[63] Zhang Lab, University of Michigan. [Online], Available: http://zhanglab.ccmb.med.umich.edu/FASTA/ , [Accessed: Feb- 2017]

[64] Shepelev V., Fedorov A. Advances in the Exon-Intron Database. Briefings in Bioinformatics 2006, 7: 178-185.

Appendix A


START
(5,145)
(41,45)
(64)
(7)
(3)
(1)
(2)
(1,2)
(99,211)
(65)
(99,60)
(56,97)
(39,52)
(23,76)
(15,31)
END


START
(5,145)
(41,68)
(13)
(9,10)
(8,40)
(10)
(16)
(65)
(55,99)
(45,61)
(99,994)
(3,30)
(1,23)
END