Scaling Up Machine Learning Algorithms to Handle Big Data

by

Khalifeh AlJadda

(Under the direction of Prof. John A. Miller and Prof. William S. York)

Abstract

Machine learning algorithms are very useful in many disciplines like speech recognition, bioinformatics, recommendations, decision making, etc. These algorithms gain more importance in the big data era due to the power of the data driven solutions. Machine learning algorithms are considered the core of data driven models. However, scalability is considered crucial requirement for all the machine learning algorithms as well as any computational model. In order to scale up the machine learning algorithms to handle big data, two basic techniques can be followed:

1. The parallelization of the existing sequential algorithms. This technique is what Apache Mahout and Apache Spark follow to scale up the machine learning algorithms.

2. Re-design the structure of existing models to overcome the scalability limitation. The result of this technique (which is more challenging) is new models which extend the existing ones, like the Continuous Bag-of-Words model.

In this thesis we apply the second technique to extend a well known machine learning technique which is Bayesian Networks to handle big data in a very efficient time and space

manner. The proposed model will lead to an easily-scalable, more readable, and expressive implementation for problems that require probabilistic solutions for massive amounts of hierarchical data. We successfully applied this model to solve three different challenging probabilistic problems, namely, multi-label classification, latent semantic discovery, and semantically ambiguous keywords discovery on massive data sets. The model was successfully tested on a single machine as well as on a Hadoop cluster of 69 data nodes.

INDEX WORDS: Massive Hierarchical Data, Distributed Algorithms, Multi-Class Classification, Mining Massive Data, Probabilistic Graphical Model, Big Data

Scaling Up Machine Learning Algorithms to Handle Big Data

by

Khalifeh AlJadda

B.S., Jordan University of Science and Technology, 2002

M.S., Jordan University of Science and Technology, 2006

A Dissertation Submitted to the Graduate Faculty

of The University of Georgia in Partial Fulfillment

of the

Requirements for the Degree

Doctor of Philosophy

Athens, Georgia

2014

Scaling Up Machine Learning Algorithms to Handle Big Data

by

Khalifeh AlJadda

Approved:

Major Professors:	John A. Miller William S. York
Committee:	Khaled M. Rasheed Krys J. Kochut

Electronic Version Approved:

Julie Coffield
Interim Dean of the Graduate School
The University of Georgia
December 2014

Dedication

To my parents, I could never have done this without your prayers, support, and constant encouragement. Thank you for teaching me to believe in Allah, in myself, and in my dreams.
To my wife and my kids, No one has ever been given more loving and happiness than I have been given by you.
To my brothers and sisters, I will be forever thankful for love and happiness you surround me with.

Love you all.

Acknowledgment

I would like to thank my professors, John A. Miller and William S. York for all what they taught me through the PhD program. I can't express how thankful I am for all you taught me. You believed in me from the very beginning and I learned from you what boost my career and make difference in my life. Many thanks to Rene Ranzinger who is very generous in terms of his time and knowledge, I'm thankful for the chance I got to work with a scientist like you. Deep thanks and hats off to Trey Grainger the director of engineering search and analytics at Careerbuilder.com. Trey believed in me like no one else, he gave me the great opportunity to apply what I learned in the graduate school, he was a great supporter to my career in data science, and he taught me how to be hardworking and to never give up. My colleagues, Melody Porterfield, Brent Weatherly, Mohammed Korayem and Camilo Ortiz, this work without you would be nothing but a dream, thank you for all your contributions. Last but not least, I would like to thank my beloved wife for her patience and unlimitted support throughout the PhD program.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the revolution caused by the big data platforms like Hadoop, collecting and analyzing ever larger datasets has become a trend in software companies. This trend leads to an increase in the importance of machine learning, which is the core of data analysis and data driven computation [11, 63]. Machine learning is widely used in many fields like bioinformatics [33], speech recognition [41], recommender systems [31], security systems [43], etc. It provides the ability to draw new insights from massive datasets, which has become one of the core targets of many domains like computational biology and searchlog analysis. Most of the machine learning algorithms belong to the 70's and 80's era when the computational resources and the datasets size were limited, so those algorithms are often not scalable enough to handle big data. Two techniques basically are applicable to scale up machine learning algorithms:

1. The parallelization of the existing sequential algorithms. This technique is what Apache Mahout and Apache Spark follow to scale up the machine learning algorithms.

2. Re-design the structure of existing models to overcome the scalability limitation. The result of this technique (which is more challenging) is new models which extend the existing ones, like the Continues Bag-of-Words model [48].

In this thesis our focus is on scaling up the well known probabilistic graphical model Bayesian Network (BN). We extended this model to a new scalable model called PGMHD which is intended to handle massive hierarchical data. The proposed model is applied successfully in bioinformatics and semantic analysis domains.

Probabilistic graphical models (PGM) refer to a family of techniques that merge concepts from graph structures and probability models [61]. They represent the conditional dependencies among sets of random variables [35]. In the age of big data, PGMs can be very useful for mining and extracting insights from large-scale and noisy data. The major challenges that PGMs face in this emerging field are the scalability and the restricted domian size (e.g., propositional domain) [32, 15]. Some extensions have already been proposed to address these challenges, such as hierarchical probabilistic graphical models (HPGM) which aim to extend the BN to work with structured domains [32, 26]. The focus of these models is to make Bayesian networks applicable to structured domains, but they do not solve the scalability issues that arise when they are applied to massive data sets.

In massive data sets which exhibit hierarchical properties, where data can be divided into several levels arranged in tree-like structures, data items in each level depend on or are influenced by only the data items in the immediate upper level. For this kind of data the most appropriate PGM to represent the probability distribution would be a Bayesian Network (BN), since the dependencies in this kind of data are not bidirectional. A Bayesian Network is considered to be feasible when it can provide a concise representation of a large probability distribution where the need cannot be efficiently handled using traditional techniques such as tables and equations [19]. Hence, with massive heirarchical data which is expected in the big data era, BN may be infeasible since it can not provide a concise representation of a large probability distribution due to the following:

1. each level represents one or more random variables, while each node in that level represents an outcome (possible value) of a random variable. So, the data grow horizontally

(number of values) much faster than vertically (number of random variables). The result is few random variables with massive values for each.

2. the structure of the network is predefined, the random variables that represent level L_i depends on the random variable that represent level $L_i - 1$, if L_i is not the root level. This makes the first phase of learning the optimal structure of BN not applicable.

For example, in the Glycan Ontology "GlycO" [65] which describes more than 1300 glycan structures (see section 3.6) whose theoretical tandem mass spectra (MS) can be predicted by GlycoWorkbench [13]. If the maximum of cleavages is set to two and the number of cross-ring cleavages is set to one, the theoretical MS^2 spectrum contains 2,979,334 ions, which themselves can be fragmented to form tens of millions of ions in MS^3. To represent this data set of only two levels of the MS data using Bayesian Network (BN) the network will be composed of two nodes, MS^1 and MS^2 with a single path $MS^1 \rightarrow MS^2$, while the Conditional Probability Table (CPT) for the MS^2 will contain 3,873,134,200 (2,979,334 $\times$ 1300) entries. For this kind of data, we propose a simple probabilistic graphical model (PGMHD) that can represent massive hierarchical data in more efficient manner. We successfully apply the PGMHD in two different domains: bioinformatics (for multi-class classification) and search log analytics (for latent semantic discovery and discovery of semantically ambiguous words).

Chapter 2

Literature Review

Probabilistic graphical models can be classified into two major categories: (1) directed graphical models, which are often referred to as Bayesian networks, or belief networks, and (2) undirected graphical models which are often referred to as Markov Random Fields, Markov networks, Boltzmann machines, or log-linear models [39]. Probabilistic graphical models (PGMs) consist of both graph structure and parameters. The graph structure represents a set of conditionally independent relations for the probability model, while the parameters consist of the joint probability distributions [61]. Probabilistic graphical models are often considered to be more convenient than numerical representations for two main reasons [52]:

1. To encode a joint probability distribution for P(x_1,...,x_n) for n propositional variables with a numerical representation, we need a table with 2^n entries. More generally, if $x_j \in \{0, ..k_j\}$, then the size of joint probability table becomes $\prod_{j=1}^{n} k_j$

2. Inadequacy in addressing the notion of independence: to test independence between x and y, one needs to test whether the joint distribution of x and y is equal to the product of their marginal probability.

PGMs are used in many domains. For example, Hidden Markov Models (HMM) are considered a crucial component for most of the speech recognition systems [41]. In bioinformatics, probabilistic graphical models are used in RNA sequence analysis [24], protein homology detection and sequence alignment [62], and for genome-wide identification [71]. In natural language processing (NLP), HMM and Bayesian models are used for part of speech (POS) tagging [16, 42]. The problem with PGMs in general, and Bayesian networks in particular, is that they may not be suitable for representing massive data due to the time complexity of learning the structure of the network and the space complexity of storing a network with a huge number of random variables with large domains. In general, finding a network that maximizes the Bayesian and Minimum Description Length (MDL) scores is an NP-hard problem [27].

2.1 Bayesian Networks

A Bayesian Network is a concise representation of a large probability distribution to be handled using traditional techniques such as tables and equations [19]. The graph of a Bayesian Network is a Directed Acyclic Graph (DAG) [35]. A Bayesian Network consists of two components: a DAG representing the structure, and a set of Conditional Probability Tables (CPTs) as shown in Figure 3.1. Each node in a Bayesian Network must have a CPT which quantifies the relationship between the variable represented by that node and its parents in the network. Completeness and consistency are guaranteed in a Bayesian Network since there is only one probability distribution that satisfies the Bayesian Network constraints [19]. The constraints that guarantee a unique probability distribution are the numerical constraints represented by CPT and the independence constraints represented by the structure itself. The independence constraint is shown in Figure 3.1. Each variable in the structure is independent of any other variables other than its parents, once its parents

are known. For example, once the information about A is known, the probability of L will not be affected by any new information about F or T, so we call L independent of F and T once A is known. These independence constraints are known as the Markovian assumptions.

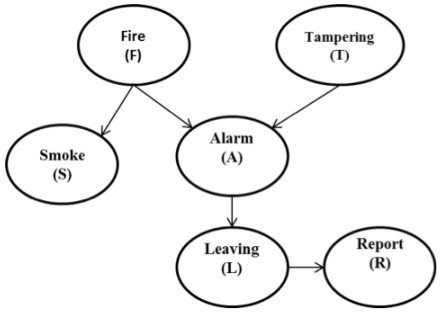


Figure 2.1: Bayesian Network

Bayesian networks are widely used for modeling causality in a formal way, for decision-making under uncertainty, and for many other applications [19].

2.2 Markov Random Fields

Markov Random Fields (MRFs), which are known also as Markov networks, are the most well-known graphical models in which the graph is undirected. In this graphical model, the random variables are represented as vertices while the edges represent dependency. However, because there is no clear causal influence from one node to the other (i.e., the link represents a direct dependency between two variables, but neither one of them is necessarily a cause for the other) the edges are undirected. In an undirected graph any two nodes without a direct link are always conditionally independent variables; whereas; any two nodes with a direct link are always dependent [35, 52]. In MRFs, the joint probability distribution can be calculated by multiplying a normalization factor by potential functions which assign positive value to a set of fully connected nodes called a clique. A clique is a fully connected subset of nodes and is associated with a non-negative potential function ϕ. Potential functions are

derived from the notion of conditional independence, so any potential function must refer only to the nodes that are directly connected (i.e., form a maximal clique). According to cliques and potential functions, the joint probability in an undirected graph shown in Figure 2.2 is calculated using the following equation:

$$p(a, b, c, d) = \frac{1}{Z}\phi_{acd}(a, c, d)\phi_{a,b}(a, b)$$

Where Z is a normalization factor that is calculated by summing or integrating the product of the potential functions:

$$Z = \sum_{a}\sum_{b}\sum_{c}\sum_{d}\phi_{a,c,d}(a, c, d)\phi_{a,b}(a, b)$$

MRFs are common in many fields like spatial statistics, natural language processing, and communication networks that have little causal structure to guide the construction of a directed graph.

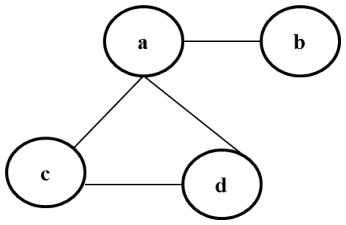


Figure 2.2: Markov Network

2.3 Hidden Markov Models

A Hidden Markov Model (HMM) is a statistical time series model which is used to model dynamic systems whose states are not observable, but whose outputs are. HMMs are widely used in speech recognition, handwriting recognition and text-to-speech synthesis [72]. HMMs

rely on three main assumptions. First, the observation at time t is generated by a process whose state S_t is hidden from observation.

Second, the state of that hidden process satisfies the Markov assumption that once the state of the system at t is known, its states and outputs at times after t are no longer dependent on states before t. In other words, the state at a specific time contains all needed information about the history of the process to predict the future of the process. Upon those assumptions, the joint probability distribution of a sequence of states and observations can be factored as follows [19, 72]:

$$P(S_{1:T}, Y_{1:T}) = P(S_1)P(Y_1|S_1)\prod_{t=2}^{T} P(S_t|S_{t-1})P(Y_t|S_t)$$

where S_t refers to the hidden state, Y_t refers to the observation at time t, and the notation $1:T$ means (1,2,..,T).

The third assumption is that the hidden state variables are discrete (i.e. S_t can take on K values). So, to define the probability distribution over observation sequences, we need to specify a probability distribution over the initial state $P(S_1)$, the $K*K$ state transition matrix defining $P(S_t|S_{t-1})$ and the output model defining $P(Y_t|S_t)$. HMMs are considered a subclass of Bayesian networks known as dynamic Bayesian networks (DBN), which are Bayesian networks that model systems that evolve over time [28].

2.4 Probabilistic Graphical Models for Hierarchical Data

Classical probabilistic graphical models require propositional domains [32]. To overcome this limitation, extensions were proposed to extend those models to non-propositional domains. A Bayesian hierarchical model has been used for natural scene categorization where it performs well on large sets of complex scenes [25]. This model has also been applied for event

recognition of human actions and interactions [50]. Another application of the hierarchical Bayesian Network is for identifying changes in gene expression from microarray experiments [9]

In [32] the authors introduced a Hierarchical Bayesian Network which extends the expressiveness of a regular Bayesian Network by allowing a node to represent an aggregation of simpler types which enables the modeling of complex hierarchical domains. The main idea is to use a small number of hidden variables as a compressed representation for a set of observed variables with the following restrictions:

1. Any parent of a variable should be in the same or immediate upper layer.

2. At most one parent from the immediate upper layer is allowed for each variable.

So, the idea is mainly to compress the observed data. Although Hierarchical Bayesian Network models extended the regular Bayesian Network to represent non-propositional domains, they have not been able to solve the issue of the scalability of Bayesian Networks for massive amounts of hierarchical data. Those models still depend on the regular learning algorithms for the optimal network structure. The learning algorithms of a Bayesian Network structure are very expensive when the random variables can take on values from large domains.

Chapter 3

Mining Massive Hierarchical Data Using a Scalable Probabilistic Graphical Model

3.1 Abstract

Probabilistic Graphical Models (PGM) are very useful in machine learning and data mining fields. However, the crucial feature in those models is the scalability. Bayesian Networks which are one of the most common PGMs used in machine learning and data mining demonstrates this limitation when the training data consists of random variables which have large sets of possible values. In the big data era, one would expect new extensions to the existing PGMs to handle the massive amount of data produced these days by computers, sensors and other electronic devices. With hierarchical data - data that are arranged in a treelike structure with several levels - one would expect to see hundreds of thousands or millions of values distributed over even just a small number of levels. When modeling this kind of hierarchical data across large data sets, Bayesian Networks may become infeasible for representing the probability distributions. In this paper, we introduce an extension to Bayesian networks to handle massive set of hierarchical data in a reasonable amount of time, and space. The

proposed model achieves perfect precision of 1.0 and high recall of 0.93 when it is used as multi-label classifier for the annotation of mass spectrometry data. On another data set of 1.6 billion search logs provided by Careerbuilder.com the model was able to predict latent semantic relationships between search keywords with accuracy up to 0.80.

3.2 Introduction

Probabilistic graphical models (PGM) consist of a structural model and a set of conditional probabilities [61, 35]. They are widely used in machine learning and data mining techniques, like classification, speech recognition [41], bioinformatics [24, 62], Natural Language Processing (NLP) [16, 42], etc. Scalability and restricted domian size (e.g., propositional domain) are the major challenges for PGMs. To overcome these challenges one would expect extensions to the existing PGMs. One extension is offered by the hierarchical probabilistic graphical models (HPGM) which aim to extend the PGM to work with more structured domains [32, 26]. However, this extension tackles the restricted domain size problem, but not scalability. For hierarchical data, where data can be divided into several levels arranged in tree-like structures, data items in each level depend on or influenced by only the data items in the upper levels. A Bayesian Network (BN) is the most appropriate PGM to represent a probability distribution, since the dependencies in this kind of data are not bidirectional. Hence, with massive heirarchical data which are expected in the big data era, BN maybe infeasible, as it may not provide a concise representation of a large probability distribution due to the following reason: each level represents a random variable, while each node in that level represents an outcome (possible value) of that random variable, so the data can grow horizontally (number of values) faster than vertically (number of random variables). Moreover, since the dependency between the random variables is pre-defined in the hierarchical

data, the structure of the network is predefined. Hence, the first phase of building Bayesian Network to find the optimal structure is not applicable

For example, in the glycan ontology "GlycO" [65] which describes 1300 glycan structures (see section 3.6) whose theoretical tandem mass spectra (MS) can be predicted by GlycoWorkbench [13]. If the maximum of cleavages is set to two and the number of cross-ring cleavages is set to one, the theoretical MS^2 spectrum contains 2,979,334 ions, which themselves can be fragmented to form tens of millions of ions in MS^3. To represent this data set of only two levels of the MS data using Bayesian Network (BN) the network will be composed of two nodes, MS^1 and MS^2 with a single path $MS^1 \rightarrow MS^2$ while the Conditional Probability Table (CPT) for the MS^2 will contain 3,873,134,200 (2,979,334 $\times$ 1300) entries. For this kind of data, we propose a simple probabilistic graphical model for massive hierarchical data (PGMHD) which we consider as an extension to the Bayesian Network (BN,) that can represent massive hierarchical data in a more efficient way. We successfully apply the PGMHD in two different domains: bioinformatics (for multi-label classification) and search log analytics (for latent semantic discovery and discover semantically ambiguous terms).

The main contributions of this paper are as follows: We propose a simple, efficient and scalable probabilistic model that extends Bayesian Network for massive hierarchical data. We successfully apply this model to the bioinformatics domain in which we automatically classify and annotate high-throughput mass spectrometry data. We also apply this model for large-scale latent semantic discovery and semantically ambiguous terms discovery using 1.6 billion search log entries provided by CareerBuilder.com, using the Hadoop Map/Reduce framework.

3.3 Background

Graphical models can be classified into two major categories: (1) directed graphical models (the focus of this paper), which are often referred to as Bayesian networks, or belief networks, and (2) undirected graphical models which are often referred to as Markov Random Fields, Markov networks, Boltzmann machines, or log-linear models [39]. Probabilistic graphical models (PGMs) consist of both graph structure and parameters. The graph structure represents a set of conditionally independent relations for the probability model, while the parameters consist of the joint probability distributions [61]. Probabilistic graphical models are often considered to be more convenient than numerical representations for two main reasons [52]:

1. To encode a joint probability distribution for P(X_1,...,X_n) for n propositional random variables with a numerical representation, we need a table with 2^n entries.

2. Inadequacy in addressing the notion of independence: to test independence between X and Y, one needs to test whether the joint distribution of x and y is equal to the product of their marginal probability.

PGMs are used in many domains. For example, Hidden Markov Models (HMM) are considered a crucial component for most of the speech recognition systems [41]. In bioinformatics, probabilistic graphical models are used in RNA sequence analysis [24]. In natural language processing (NLP), HMM and Bayesian models are used for part of speech (POS) tagging [16]. The problem with PGMs in general, and Bayesian networks in particular, is that they may not be suitable for representing massive data due to the time complexity of learning the structure of the network and the space complexity of storing a network with a huge number of random variables or random variables taking in many values. In general, finding a network that maximizes the Bayesian (maximizes the posterior probability) and

Minimum Description Length (MDL, preffer a simple BN over a complex one) scores is an NP-hard problem [27].

Bayesian Networks

A Bayesian Network is a concise representation of a large probability distribution to be handled using traditional techniques such as tables and equations [19]. The graph of a Bayesian Network is a Directed Acyclic Graph (DAG) [35]. A Bayesian Network consists of two components: a DAG representing the structure (as shown in Figure 3.1), and a set of Conditional Probability Tables (CPTs). Each node in a Bayesian Network must have a CPT which quantifies the relationship between the variable represented by that node and its parents in the network. Completeness and consistency are guaranteed in a Bayesian Network since there is only one probability distribution that satisfies the Bayesian Network constraints [19]. The constraints that guarantee a unique probability distribution are the numerical constraints represented by CPT and the independence constraints represented by the structure itself. The independence constraints is shown in Figure 3.1. Each variable in the structure is independent of any other variables other than its parents, once its parents are known. For example, once the information about A is known, the probability of L will not be affected by any new information about F or T, so we call L independent of F and T once A is known.

Bayesian networks are widely used for modeling causality in a formal way, for decision-making under uncertainty, and for many other applications [19].

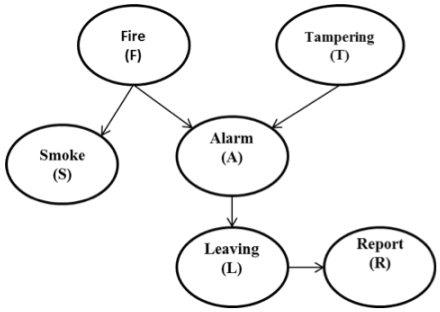


Figure 3.1: Bayesian Network

3.4 Related Work

Our research is related closely to Bayesian Network classifiers. In this section, we review different forms of Bayesian Network classifiers to understand how the PGMHD extends BN in a way different than the existing models. We cover the following BN classifiers:

1. Naive Bayes Classifier (NB).
2. Selective Naive Bayes (SNB)
3. Tree Augmented Naive Bayes (TAN).
4. Hidden Naive Bayes (HNB).

On the other hand, we applied PGMHD to other data mining problems, namely, latent semantic discovery between search keywords in users search logs and discover the semantically ambiguous keywords by analyzing users' search logs.

Naive Bayes (NB)

Naive Bayes is the simplest form of the BN classifiers and the most common one. This classifier is based on an assumption that all the features are independent given the class.

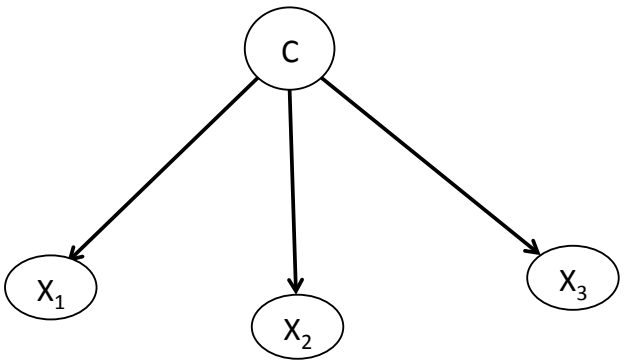


Figure 3.2: Naive Bayes

Figure 3.2 shows an example of NB. A NB classifier is defined as follows: The conditional probability of class C given input vector $\mathbf{X}$ is proportional to the prior probability times the product of the conditional probabilities,

$$P(C|\mathbf{X}) \propto P(C) \prod_{j=1}^{n-1} P(X_j|C)$$

where $\mathbf{X} = (X_1, \ldots X_{n-1})$, $P(C)$ is the prior probability of class C and $P(X_j|C)$ is the conditional probability of feature/variable X_j given class C. The value of C that maximizes the right hand side is chosen. A Naive Bayes classifier performance depends upon the quality of the predictor features where the performance is improved when the predictor features are relevant and non redundant.

Selective Naive Bayes

One may improve the performance of NB classifiers by selecting the predictive features that are relevant and not redundant. Selective Naive Bayes (SNB) [7] addresses this by icluding

a feature subset selection problem. Let us define $\mathbf{X}_F$ as the projection of $\mathbf{X}$ onto a selected feature subset $F \subseteq \{1, 2, \ldots n\}$. The classification equation now becomes

$$P(C|\mathbf{X}_F) \propto P(C) \prod_{j \in F} P(X_j|C)$$

Tree Augmented Naive Bayes

A Tree Augmented Naive Bayes (TAN) is a form of Bayesian Network classifier that extends NB classifiers by allowing each attribute to have at most one attribute parent in addition to its class parent. This extension tends to represent the fact that in some cases there is dependency or influence between features in a way that a value of a feature X_j depends on a value of feature $ap(X_j)$. TAN classifier is defined as follows:

$$P(C|\mathbf{X}) \propto P(C) \prod_{j=1}^{n-1} P(X_j|ap(X_j), C)$$

where $ap(X_j)$ is the attribute parent of X_j. Figure 3.3 shows an example of a TAN, in which X_1 is the ap for both X_2 and X_3. C is a parent to X_1, X_2 and X_3.

Hidden Naive Bayes

A Hidden Naive Bayes (HNB), depicted in Figure 3.4, is another extension of NB classifiers [73]. In this extension, each attribute/variable X_j gets a hidden parent $hp(X_j)$ to integrate the influences from one or more of the other attributes/variables. This extension makes the model more expressive than TAN which allows at most one attribute parent while it ignores the others. The hidden parent in HNB integrates a weighted influence of other attributes in one node. The definition of the HNB classifier is as follows:

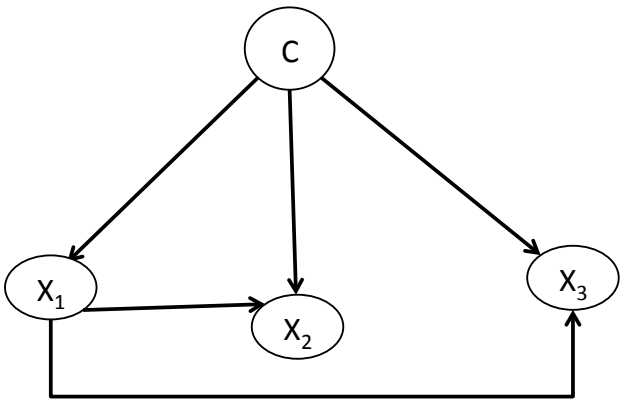


Figure 3.3: Tree Augmented Naive Bayes

$$P(C|\mathbf{X}) \propto P(c) \prod_{j=1}^{n-1} P(X_j|hp(X_j), C)$$

where,

$$P(X_j|hp(X_j), C) = \sum_{i=1, i\neq j}^{n-1} w_{ji} * P(X_j|X_i, C)$$

where w_{ji} is a weight that reflects the influence of attribute X_i on attribute X_j.

3.5 Probabilistic Graphical model for Massive Hierarchical Data Problems (PGMHD)

In this section, we describe PGMHD. We discuss the structure of the model, its learning algorithm, and how it extends Bayesian Networks. PGMHD includes a post-processing step on top of ML^{2}BN, a restricted form of Bayesian Network suitable for solving very large problems.

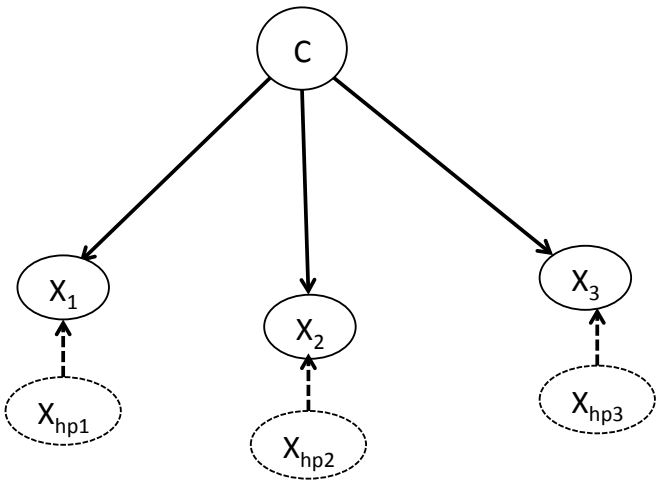


Figure 3.4: Hidden Naive Bayes

Model Structure

The structure of a model is very inportant in determining both how effectively and how efficiently a model can be run. As mentioned Bayesian Networks are based on Directed Acyclic Graphs, since one does does not want to consider a varaibles dependency with itself. The concept of Directed Acyclic Graph may be defined as follows:
Definition: Directed Acyclic Graph (DAG) is a 2-tuple $G(V, E)$ where

- V = set of vertices
- $E \subset V \times V$
- there is no path that contains the same vertex twice (no cycles)

In this work, to increase the suitable for handling very large networks, a further restriction is applied. Many domains have a natural multi-level structure that can be used to reduce complexity. The concept of a Multi-Level DAG has been defined by [4].

Definition: Multi-level DAG (MLDAG) is a 3-tuple $G(V, E, L)$ where

- V = set of vertices
- $E \subset V \times V$
- there is no path that contains the same vertex twice (no cycles)
- $L = \{L_1, \ldots L_m\}$ is a set of m levels
- V is partitioned into $\{L_1, \ldots L_m\}$
- $e = (u, v) \in E \Rightarrow u \in L_{i-1}$ and $v \in L_i$ for some i

The partitioning implies that $L_i \cap L_j = \phi$ for $i \neq j$.

3.5.1 Definitions of Bayesian Networks

In this work, the attributes/variables are (Finite) Random Variables. Although the development could be extended to discrete (including countably infinite) and continuous cases, finite works for the real-world applications we are addressing. The concept of (Finite) Random Variable has been defined by [49].

Definition: Finite Random Variable $X : \Omega \rightarrow \{x_1, \ldots x_k\}$ with an associated probability measure P where

- Ω = set of outcomes of an experiment
- $\{x_1, \ldots x_k\}$ is a finite set of real numbers
- $P(X = x_i) \in [0, 1]$ is the probability of X taking the value x_i

Bayesian Networks have been discussed earlier and are now more formally defined. Their utility stems from the fact that the joint probability of the random variables can be factored,

greatly reducing the complexity and storage requirements. The concept of Bayesian Network has been defined by [52].

Definition: (Finite) Bayesian Network is a 3-tuple $\mathrm{BN}(G, X, f)$ where

- G is a DAG
- $X = \{X_1, \ldots X_n\}$ is a set of finite random variables
- $f : G.V \to X$ is a bijective function mapping vertices to random variables
- X_i directly depends on X_j $(i \neq j)$ iff $(f^{-1}(X_i), f^{-1}(X_j)) \in G.E$

When X_i is directly dependent on X_j, X_j may be referred to as the parent of X_i. A random variable is independent of all random variables except its ancestors. The parents of a random variable X_i are given by the following function.

$$pa : X \to 2^X$$

The joint probability is given by

$$P(X_1, \ldots X_n) = \prod_{i=0}^{n} P(X_i | pa(X_i))$$

Note, the above definition can be generalized to discrete or continuous cases. An important special case of Bayesian Network is Naive Bayes [59]

Definition: (Finite) Naive Bayes is a 3-tuple $\mathrm{NB}(G, X, f)$ where

- X_n is distinguished, removed and re-labeled as C
- for $i \in \{1, \ldots n-1\}$, X_i only depends on C

This causes G to become a one level, single rooted tree. Now by Bayes Theorem, the conditional probability of C given the rest of of the random variables is as follows.

$$P(C|X) \;=\; \frac{P(X|C)P(C)}{P(X)}$$

Since the values for X for a given input are invariant, we may write

$$P(C|X) \;\propto\; P(X|C)P(C)$$

Since the X_i's are independent of each other, the conditional probabilities may be multiplied.

$$P(C|X) \;\propto\; P(C)\prod_{i=0}^{n-1} P(X_i|C)$$

This can be generalized to Multi-label Naive Bayes, by distinguishing k of the random variable and re-labeling them $C_1, \ldots C_k$.

Combining a Multi-level DAG with a Multi-label Bayesian Network we obtain to following.

Definition: (Finite) Multi-label, Multi-level Bayesian Network is a 3-tuple $\mathrm{ML^2BN}(G, X, f)$ where

- G is an MLDAG
- $X \;=\; \{X_1, \ldots X_n\}$ is a set of finite random variables
- $f : G.V \to X$ is a bijective function mapping vertices to random variables
- X_i directly depends on X_j $(i \neq j)$ iff $(f^{-1}(X_i), f^{-1}(X_j)) \in G.E$

The last k random variables are distinguished and re-labeled to form the top level of the MLDAG.

$$L_1 \;=\; \{f^{-1}(C_1), \ldots f^{-1}(C_k)\}$$

The rest of the random variables are partitioned among the remaining levels.

$$f^{-1}(X_j) \in L_i \text{ for some } j \in \{1, \ldots n-k\} \text{ and } i \in \{2, \ldots m\}$$

The joint probability is given by

$$P(X_1, \ldots X_n) \;=\; \prod_{i=0}^{n} P(X_i|pa(X_i))$$

In this case, if $f^{-1}(X_j) \in L_i,$ then $f^{-1}(pa(X_j)) \subseteq L_{1-1}$. This uses the natural extension of a function to apply to a set of values from its domain.

A special case of ML^{2}BN occurs when there is a single label and only 2 levels. In this case, the joint probability reduces to

$$P(X_1, \ldots X_{n-1}, C) \;=\; P(C) \prod_{i=0}^{n-1} P(X_i|C)$$

since $pa(C) = \phi$ and $pa(X_i) = C$. Consequently,

$$P(C|X_1, \ldots X_{n-1}) \;=\; \frac{P(X_1, \ldots X_{n-1}, C)}{P(X_1, \ldots X_{n-1})}$$

Factoring out the common $P(X_1, \ldots X_{n-1})$ as before and substituting for the joint probability, we obtain

$$P(C|X_1, \ldots X_{n-1}) \propto P(C) \prod_{i=0}^{n-1} P(X_i|C)$$

which is the expression for Naive Bayes.

A second special case occurs when there are two levels and k labels. In this case, the joint probability for given label/class C_j is given by

$$P(X_1, \ldots X_{n-k}, C_j) \;=\; P(C_j) \prod_{i=0}^{n-k} P(X_i|pa(X_i))$$

where $pa(X_i) \subseteq L_1$. This corresponds to Multi-label Naive Bayes. The effectiveness of the model can be improved by only including the children ($ch = pa^{-1}$) of C_j, along the lines of what is done in Selective Naive Bayes [44]. This yields

$$P(X_1, \ldots X_{n-k}, C_j) \;=\; P(C_j) \prod_{X_i \in ch(C_j)} P(X_i|pa(X_i))$$

Letting $ch(C_j) = \{X_{j1}, \ldots X_{jl}\}$, we obtain the conditional probability of of label/class C_j given the relevant input.

$$P(C_j|X_{j1}, \ldots X_{jl}) = \frac{P(C_j)}{P(X_{j1}, \ldots X_{jl})} \prod_{i=1}^{l} P(X_{ji}|pa(X_{ji}))$$

In the common case when C_j is binary, the conditional probability for each C_j can be calculated and the top few selected as the labels.

The conditional probabilities $P(X_{ji}|pa(X_{ji}))$ can be estimated and stored in Conditional Probability Tables (CPTs) as is common practice for general Bayesian Networks. For cases when the CPTs become too large, we can use the following two approximations:

1. Replace $pa(X_{ji})$ with C_j and C_h, where C_h is the class with the highest correlation/Pointwise Mutual Information (PMI) to X_{ji}. In case $h = j$, let C_h have the second highest correlation. In the binary case, the CPT for X_{ji} will appear as follows:

X_{ji}	C_j	C_h	$P(X_{ji})$
0	0	0	p_1
0	0	1	p_2
0	1	0	p_3
0	1	1	p_4
1	0	0	p_5
1	0	1	p_6
1	1	0	p_7
1	1	1	p_8

2. Simply replace $pa(X_{ji})$ with C_j.

We now consider the case when the number of levels m is greater than two. For a three level ML^{2}BN, the conditional probability of C_j given input for levels 2 $(X_{j1}, \ldots X_{jl})$ and 3 $(Y_{j1}, \ldots Y_{jg})$, $P(C_j|X_{j1}, \ldots X_{jl}, Y_{j1}, \ldots Y_{jg})$ is given by

$$\frac{P(C_j)}{P(X_{j1}, \ldots X_{jl}, Y_{j1}, \ldots Y_{jg})} \Big[\prod_{i=1}^{l} P(X_{ji}|pa(X_{ji})) \Big] \Big[\prod_{i=1}^{g} P(Y_{ji}|pa(Y_{ji})) \Big]$$

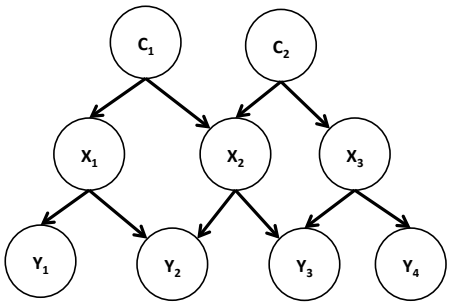


Figure 3.5: ML^{2}BN

The abstract example below illustrates this for a small problem (see Figure 3.5), $P(C_1|X_1, X_2, Y_1, Y_2, Y_3)$ is given by

$$\frac{P(C_1)}{P(X_1, X_2, Y_1, Y_2, Y_3)} \Big[\prod_{i=1}^{2} P(X_i|pa(X_i)) \Big] \Big[\prod_{i=1}^{3} P(Y_i|pa(Y_i)) \Big]$$

A similar expression can be developed for C_2. The same two approximations can be applied as before. Although the model can be extended in a similar fashion to include more levels, all the real-world case studies considered in this paper have either 2 or 3 levels.

Probabilistic-based Classification

In this section, we will use a classification score based on approximation scheme 2 to estimate conditional probabilities. For a random variable at level $i \in \{2, \ldots m\}$, namely $X_{ij} \in L_i$, where X_{ij} is the jth random variable at level i, we calculate a *classification score* $Cl(C'_k|X_{ij})$ for X_{ij} given its primary parent $C'_k \in L_{i-1}$. It is used to estimate the conditional probability $P(C'_k|X_{ij})$. The notation C'_k is used to denote a parent and when it is at level 1, it will represent class C_j as denoted previously. Let

$$f(C'_k, X_{ij}) = Frequency\, of\, co-occurrence\, of\, C'_k\, and\, X_{ij}$$

$$\mathrm{Cl}(C'_k|X_{ij}) \quad := \quad \frac{f(C'_k, X_{ij})}{in(X_{ij})}$$

The classification score is the ratio of the co-occurrence frequency of C'_k and X_{ij} divided by the total occurrence of X_{ij}. The total occurrence of X_{ij} is calculated by summing up the frequencies of the co-occurrence of X_{ij} and all its parents.

$$in(X_{ij}) := \sum_{C \in \mathrm{pa}(X_{ij})} f(C, X_{ij}), \qquad \forall X_{ij} \in V,$$

Now we show why this is a good estimator for the conditional probability of C'_k given X_{ij}, based on the definition of conditional probability.

$$P(C'_k|X_{ij}) = \frac{P(C'_k, X_{ij})}{P(X_{ij})}$$

The ratio of the co-occurrence of C'_k and X_{ij} over the total occurrence of X_{ij} estimates the ratio of the joint probability of C'_k and X_{ij} over the probability of X_{ij}.

To deal with the problem of zero frequency, we utilize m-estimates [38]

$$\frac{f(C'_k) + mp}{\#classes + m} \tag{3.1}$$

where $f(C'_k)$ is the frequency of class C'_k, p is a parameter often used as the prior estimate for the probability of C'_k, and m is a second parameter which is called an equivalent sample size. Our default settings are $p = 0.1$ and $m = 1$.

In some use cases, post processing is needed to reduce the effects of noise or to handle special cases that lead to incorrect classification. In this paper, we will present an example of post processing of the calculated classification score in the experiments and results section.

Probabilistic-based Similarity Scoring

Fix a level $i \in \{2, \ldots, m\}$, and let $X, Y \in L_2 \times \cdots \times L_m$ be identically distributed random variables. We define the *probabilistic-based similarity score CO (Co-Occurrence)* between two independent siblings $X_{ij}, Y_{ig} \in L_i$ by computing the conditional joint probability

$$\mathrm{CO}(X_{ij}, Y_{ig}) := P(X_{ij}, Y_{ig} | \mathrm{pa}(X_{ij}) \cap \mathrm{pa}(Y_{ig}))$$

as

$$\mathrm{CO}(X_{ij}, Y_{ig}) = \prod_{C'_k \in \mathrm{pa}(X_{ij}) \cap \mathrm{pa}(Y_{ig})} P(X_{ij}|C'_k) P(Y_{ig}|C'_k),$$

where $P(X_{ij}|C'_k) = \frac{P(C'_k, X_{ij})}{P(C'_k)}$ for every $(X_{ij}, C'_k) \in L_{i-1} \times L_i$.

Given $out(C'_k)$ as the total number of occurrences of C'_k, $f(C'_k, X_{ij})$ as the frequency of co-occurrence of C'_k with X_{ij} we can naturally estimate the joint probabilities $P(X_{ij}, C'_k)$ with $\hat{p}(X_{ij}, C'_k)$ defined as

$$\hat{p}(X_{ij}, C'_k) := \frac{f(C'_k, X_{ij})}{out(C'_k)}.$$

Hence, we can estimate the correlation between X_{ij} and Y_{ig} by estimating the probabilistic similarity score $\mathrm{CO}(X_{ij}, Y_{ig})$.

Progressive Learning

PGMHD is designed to allow progressive learning (shown in Algorithm 2). Progressive learning is a learning technique that allows a model to learn gradually over time. Training data does not need to be given at one time to the model. Instead, the model can learn from any available data and integrate the new knowledge incrementally. This learning technique is very attractive in the big data age for the following reasons:

1. Training the model does not require processing all data upfront

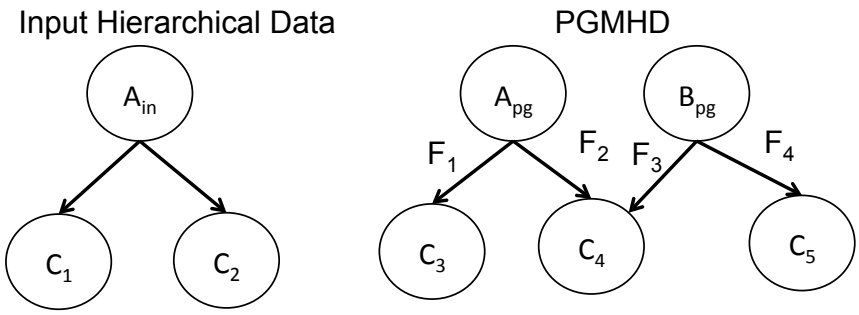


Figure 3.6: Input hierarchical data and PGMHD

2. It can easily learn from new data without the need to re-include the previous training data in the learning.

3. The training session can be distributed instead of doing it in one long-running session.

4. It supports recursive learning which allows the results of the model to be used as new training data, provided they are judged to be accurate by the user.

PGMHD an extension to NB

PGMHD extends NB in different directions to improve its scalability and ability to handle massive hierarchical data as following:

1. Enable multi-label classification.

2. Enable multi-level representation of the predictive features.

3. Enable lazy classification.

The first dimension PGMHD extends is the multi-label classification. Our model allows more than one class to be in the root level of the classifier where any instance can be classified to more than one class. The second dimension of this extension is the multi-level classification

```
Data: Input Hierarchical Data
Result: PGMHD Instance
begin
    currentLevel = 0
    while currentLevel < maxInputLevel do
        foreach inputNode ∈ pgmhd(currentLevel) do
            if inputNode exists in PGMHD then
                get pgmhdNode where pgmhdNode.data = inputNode.data
                pgmhdNode.frequency+ = 1
            else
                pgmhdNode = newnode
                pgmhdNode.frequency = 1
            end
            childrenLevel = currentLevel + 1
            foreach inputChildNode ∈ inputNode.children do
                foreach pgmhdChildNode ∈ pgmhdNode.children do
                    if inputChildNode.data = pgmhdChildNode.data then
                        edge = edge(pgmhdNode, pgmhdChildNode)
                        edge.frequency+ = 1
                    else
                        if childNode ∈ pgmhd(childrenLevel) then
                            pgmhdChildNode = node where
                            node.data = childNode.data
                            edge = createNewEdge(pgmhdNode, pgmhdChildNode)
                            edge.frequency = 1
                        else
                            pgmhdChildNode = newNode
                            pgmhdChildNode.data = child
                            pgmhdChildNode.frequency = 1
                            edge = createNewEdge(pgmhdNode, pgmhdChildNode)
                            edge.frequency = 1
                        end
                    end
                end
            end
        end
        currentLevel = currentLevel + 1
    end
end
```

Algorithm 1: Learning Algorithm for PGMHD. *currentLevel* represents the current level in the input hierarchical data, we start with level_0. *maxInputLevel* is the highest level in the input hierarchical data. In Figure 3.6 A_{in} is an *inputNode*, while A_{pg} is the *pgmhdNode*. C_1 is *inputChildNode*. C_3 is $PgmhdChildNode$. F_1 is $edge.Frequency$. $C_1, C_2 \in inputParentNode.children$. $C_3, C_4, C_5 \in PgmhdParentNode.children$

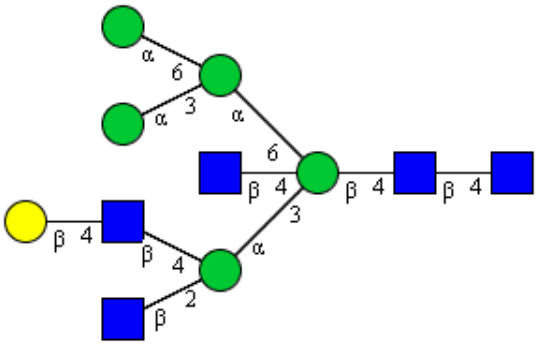


Figure 3.7: Glycan structure in CFG format. The circles and squares represent the monosaccharides which are the building blocks of a glycan while the lines are the linkages between them

which allows the classifier to represent the predictive features in m levels instead of only 2 levels as in the regular NB. This extension allows the hierarchical modeling to preserve the structure of the data, which our experiments show its importance in improving the quality of the classification. The last dimension of this extension is the lazy classifier against the eager NB. PGMHD is considered a lazy classifier since the calculation of the classification score of a new instance is all done during the classification process not like NB where all the CPT are pre-calculated and stored. This extension makes the PGMHD suitable for the progressive learning which is very important for scalability.

3.6 Experiments and Results

Glycans (Figure 3.7) are the third major class of biological macro-molecules besides nucleic acids and proteins [5]. Glycomics refers to the scientific attempts to characterize and study glycans, as defined in [5] or an integrated systems approach to study structure-function relationships of glycans as defined in [57].

Mass spectrometry (MS) is an analytical technique used to identify the composition of a sample [45]. Although (MS) has become the major analytical technique for glycans, no general method has been developed for the automated identification of glycan structures using MS and tandem MS data. MS^n refers to the sequence of MS selection with some form of fragmentation, it is also called MS/MS. The relative ease of peptide identification using tandem MS is mainly due to the linear structure of peptides and the availability of reliable peptide sequence databases. In proteomic analyses, a mostly complete series of high abundance fragment ions is often observed. In such tandem mass spectra, the mass of each amino acid in the sequence corresponds to the mass difference between two high-abundance peaks, allowing the amino acid sequence to be deduced. In glycomics MS data, ion series are disrupted by the branched nature of the molecule, significantly complicating the extraction of sequence information. In addition, groups of isomeric monosaccharides commonly share the same mass, making it impossible to distinguish them by MS alone. Databases for glycans exist but are limited, minimally curated, and suffer badly from pollution from glycan structures that are not produced in nature or are irrelevant to the organism of study. PGMHD attempts to employ machine learning techniques (mainly probabilistic-based multi-label classification) to find a solution for the automated identification of glycans using MS data.

We recently implemented the Glycan Elucidation and Annotation Tool (GELATO), which is a semi-automated MS annotation tool for glycomics integrated within our MS data processing framework called GRITS (http://www.grits-toolbox.org/). Figures 3.8, and 3.9 show screen shots from GELATO for annotated spectra. Figure 3.8 shows the MS profile level and Figure 3.9 shows the annotation of MS^2 peaks using fragments of a selected candidate glycan for annotation of the MS^1 data. The output GELATO represent all the possible annotations to the given spectra. The user may select a subset of those possible annotations as the right ones, but then he/she needs a smarter tool that can learn the right selection and

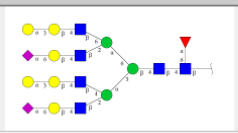
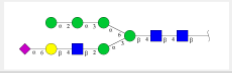
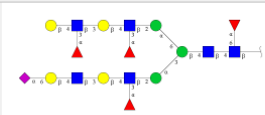
Scan #	Peak Charge	Peak Intensity	Peak m/z	Cartoon	Feature m/z	Glycan Id
117	3	144887.0	1085.923		1085.5136	GOG166
118	-1	249107.0	812.4		812.0444	GOG120
119	2	79236.5	1023.9		1023.7372	GOG516

Figure 3.8: MS^1 annotation using GELATO. Scan is the ID number of the scan in the MS file, peak charge is the charge state of that peak in the MS file, peak intensity represents the abundance of an ion at that peak, peak m/z is the mass over charge of the given peak, cartoon is the annotation of that peak (glycan) in CFG format, feature m/z is the mass over charge for the glycan, and glycanID is the ID of the glycan in the Glycan Ontology (GlycO).

eliminate the incorrect ones in the future. PGMHD is successfully applied for that purpose as we show in this section.

To represent the MS data annotation using PGMHD, each annotation of MS^1 data (which is a glycan) is represented as a node in the top-layer of PGMHD. All the fragments generated by that glycan and used to annotate peaks in MS^2 are represented by nodes in the lower layer and connected by edges with the parent node in the upper layer and this can be extended until MS^n. Each fragment at level MS^i is represented by a node in layer L_{i-1} and connected by an edge with its parent node at layer L_{i-2}. The edge's weight represents the co-occurrence frequency between a child and a parent, storing frequencies rather than probabilities facilitates progressive learning. Figure 3.10 shows the PGMHD for MS data with three levels (MS^1, MS^2, and MS^3) in these figures. As shown in the model, three layers are created: one for the MS^1 level, a second one for the MS^2 level, and a third for MS^3. Several different nodes at the MS^1 level can be annotated with the same fragment ion

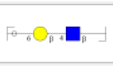
Peak Intensity	Peak m/z	Cartoon	Feature m/z	Glycan Id
13.1097	398.3249		398.1693	GOG166
5.2044	445.3975		445.1952	GOG166
13.2528	472.3444		472.2061	GOG166
13.2528	472.3444		472.2061	GOG166
13.2528	472.3444		472.2061	GOG166
8.7515	474.3912		474.2217	GOG166
2.8145	690.4586		690.3215	GOG166
2.8145	690.4586		690.3215	GOG166

Figure 3.9: Fragments of a selected glycan at the MS^2 level. Each ion observed in MS^1 is selected and fragmented in MS^2 to generate smaller ions, which can be used to identify the glycan structure that most appropriately annotates the MS^1 ion. Theoretical fragments of the glycan structure that had been used to annotate the MS^1 spectrum are used to annotate the corresponding MS^2 spectrum.

at the MS^2 level, so MS^2 nodes can have several parents. The frequency values are shown on the edges.

Experiment Setup

We annotated 3314 MS scans of pancreatic cancer samples using GELATO. Then an expert manually approved 1990 scan annotations which we used to train and test PGMHD. We split this data set into a training data and test data sets. The size of the training data is 1779 scans and 121 scans for a testing. We trained PGMHD and set of the best classifiers including Naive Bayes [59], SVM [18], Decision Tree [55], K-NN [3], Neural Network [69], Radial Basis Function network (RBF Network) [53] and Bayesian Network [51] from Weka [34]. Then we provide the test list to each classifier to predict the best glycan that annotates

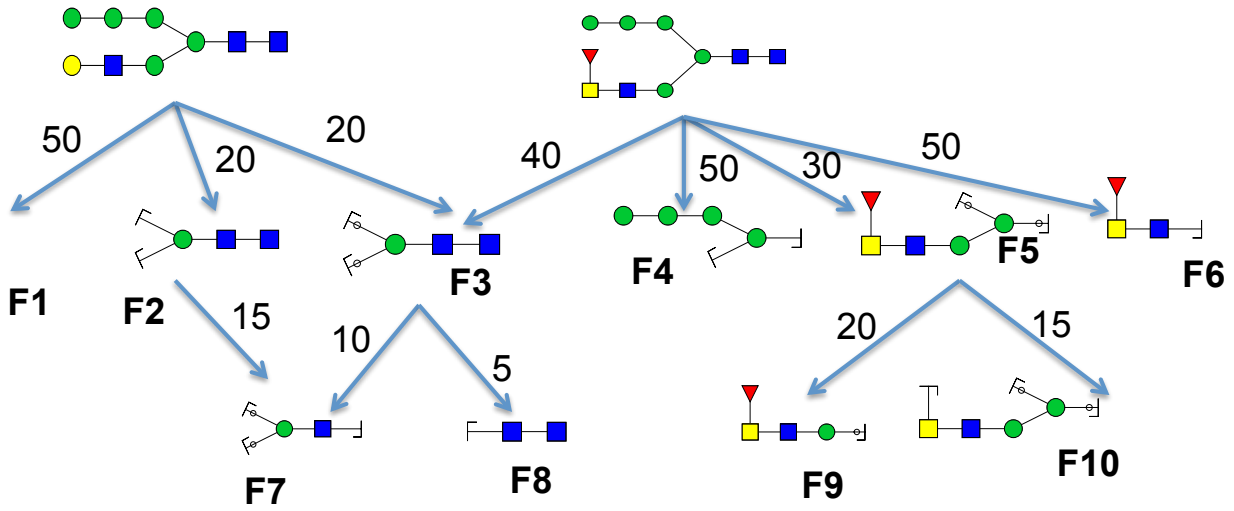


Figure 3.10: PGMHD representing MS annotations. The root nodes are the glycans that annotate the peaks at MS^1 level, while the level 2 and 3 nodes are the glycan fragments that annotate the peaks at MS^2 and MS^3 level respectively and the edges represent dependency associating the glycans with their MS^2 fragments and MS^2 with their MS^3 fragments. The number on each edge represents the frequency of seeing the destination of the edge caused by or depends on its source in the training data

the scans in the test set. We also used Mulan [67], which is a Java library that extends Weka classifiers to handle multi-label classification problems. Also, we applied the m-estimate as a probability estimation technique To help PGMHD overcome the common problem for any Beysian model which is the zero-frequency problem [38]. In some special cases a glycan which may annotate one or few peaks with high co-occurrence may get higher score than the one which may annotate more peaks but with lower co-occurrence. In this case the glycan with higher co-occurrence will be given higher score, however we prefer the one which may annotate more peaks even with less co-occurrence. To handle this special case we applied a post processing technique which multiply the calculated classification score with the fraction of number of annotated peaks to the total number of peaks in the input spectra. We ran an experiment with then without post processing and our result shows that the post

processing has a very tiny effect in our results, the recall with post processing was 0.9273 while without post processing it was 0.9272. Figure 3.11 shows the precision and recall for the different classifiers compared to PGMHD after we used Mulan for multi-label classification and m-estimate for PGMHD. Another important aspect in our experiment besides accuracy is the scalability. In order to measure the scalability of PGMHD compared to the other classifiers we measure the space and time complexity. Figure 3.12 shows how PGMHD was the fastest model in the training phase, however it was not the best in the classification time as shown in Figure 3.13, even though it got the third best time. Most important is the space complexity used by each model which is shown in figure 3.14. The memory usage which is a very important aspect in the scalability shows that PGMHD is much better than all the other classifiers especially the Bayesian ones. Due to the difficulty of getting more manually curated MS annotations dataset for testing the scalability of our model in comparison to other machine learning models, we synthesized a dataset using GELATO. To synthesize a dataset with massive number of MS annotations we used GELATO to generate all the possible annotations for the MS experiments which were manually curated before. We assume that all the generated annotations by GELATO are valid and correct annotations. Our focus in this part of the experiment is the scalability not the accuracy since the accuracy was already tested using the manually curated dataset. The new dataset includes 6776 instance for training and 392 instance for testing. The number of features is 2952 while the number of classes is 1340. As a result to this extension in the training data, Baysian Network classifer, K-NN, and RBF ran out of memory which means they can not handle this dataset in 4 GB of main memory. On the other hand, PGMHD used only 160 MB to represent this dataset in memory. Figure 3.15 shows the memory usage of the models which scale successfully to handle the new dataset.

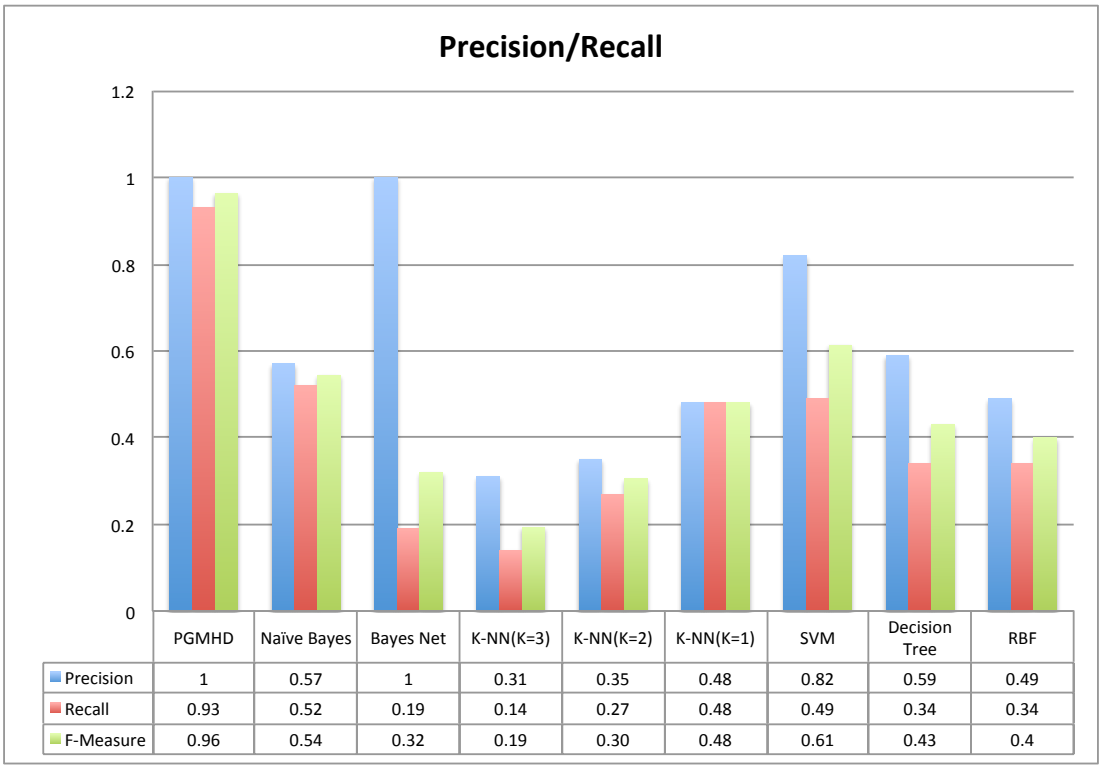


Figure 3.11: Precision and Recall after the Multi-label classification. Also, PGMHD was applied with the m-estimate where $m = 1$ and $p = 0.1$.

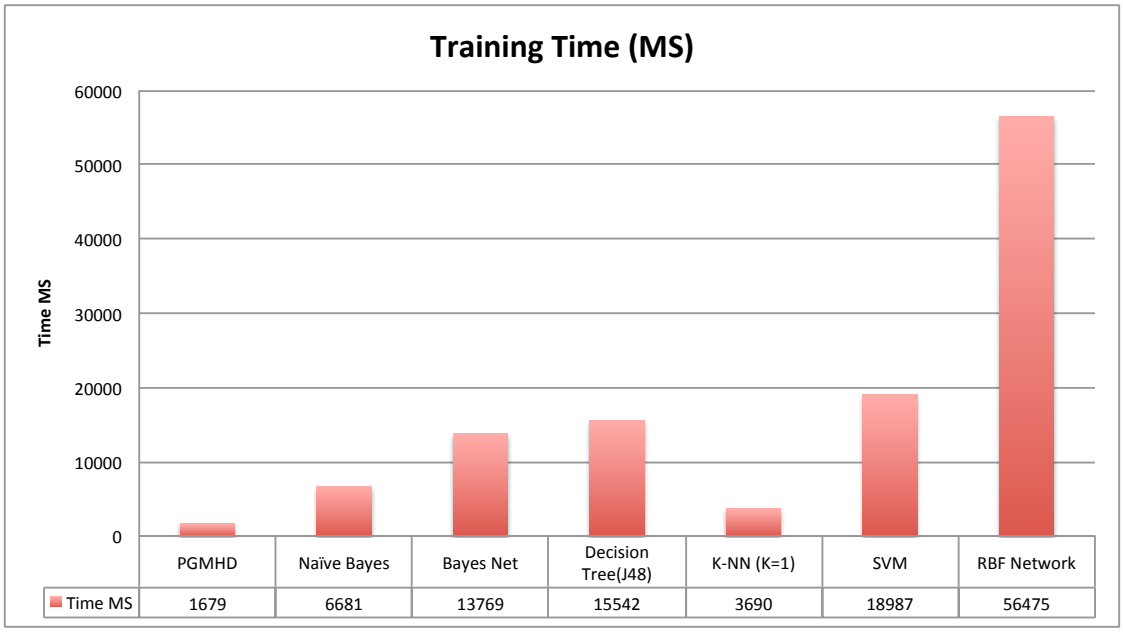


Figure 3.12: Training time for different classifiers. Lazy classifiers (PGMHD, and K-NN) are much faster in the training phase due to the fact that no complicated calculation is required.

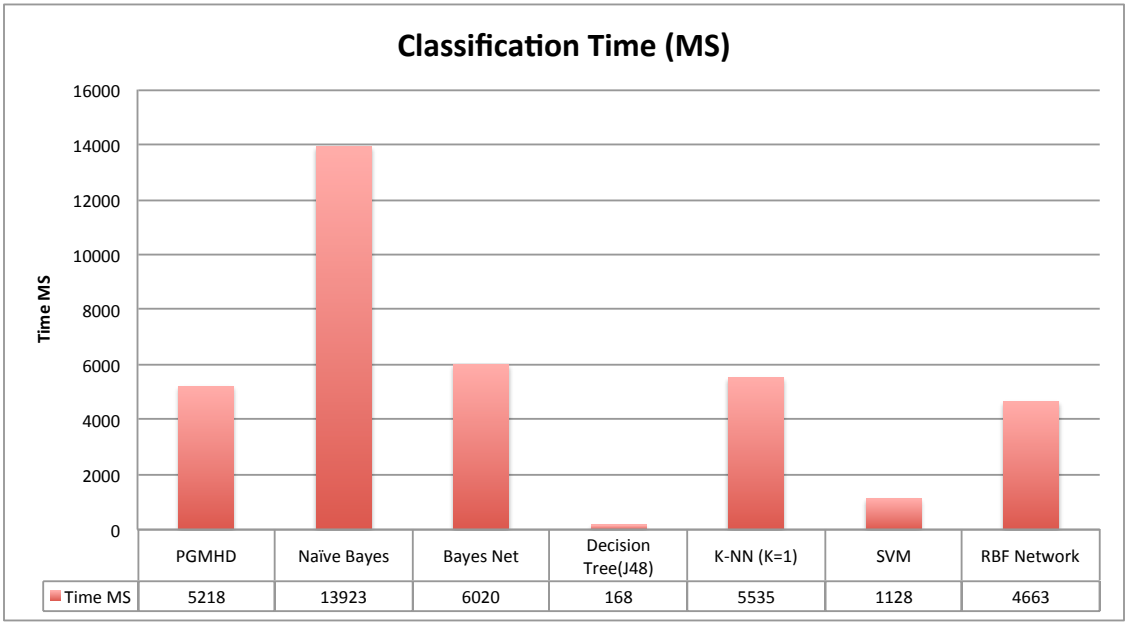


Figure 3.13: Classification time for different classifiers. The eager classifiers (Decision Tree, SVM, and RBF) are faster than the lazy ones sue to the fact that the complicated computation are done during the training phase which cause the classification time to be faster. Naive Bayes and Bayesian Network didn't do well due to the multi-label classification which they are not suitable for.

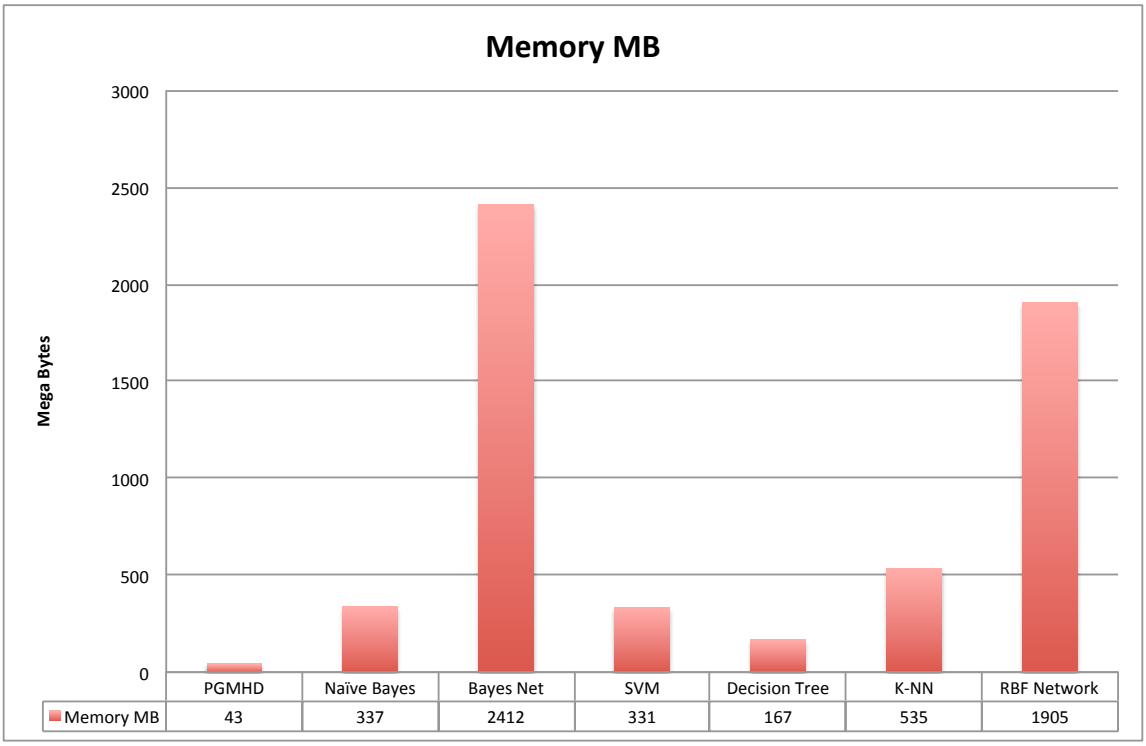


Figure 3.14: Memory usage by each model in MB for a dataset of 1779 instances annotated by 468 glycans (classes).

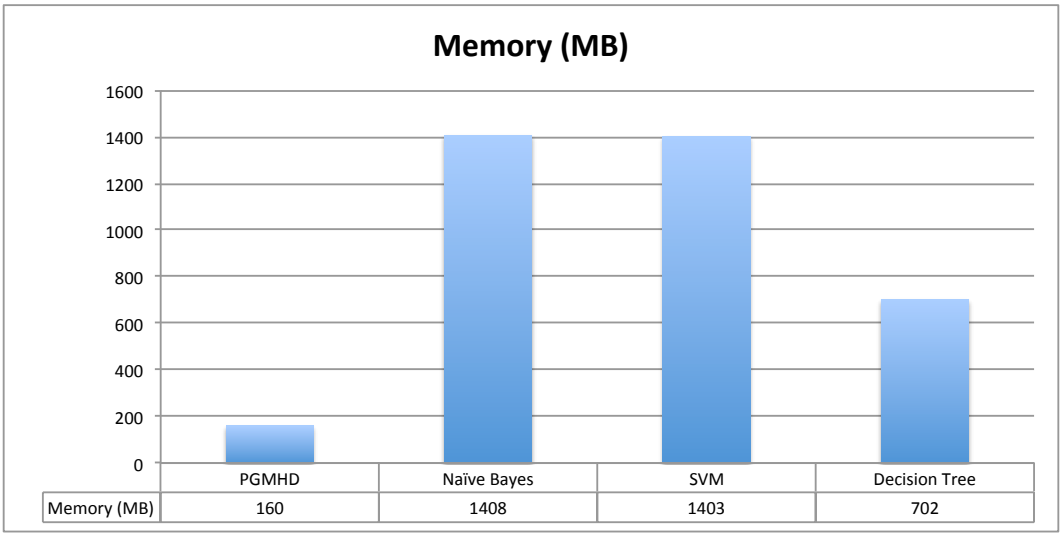


Figure 3.15: Memory usage by each model in MB for a training dataset of 6776 instances, 1640 features, and annotated by 1340 glycans (classes).

Semantically Related Keywords in Search Logs

Semantic similarity, is a metric that is defined over documents or terms in which the distance between them reflects the likeness of their meaning [36], It is widely used in Natural Language Processing (NLP) and Information Retrieval (IR) [47]. Generally, there are two major techniques used to compute the semantic similarity: one is computed using a semantic network (Knowledge-based approach) [12], and the other is based on computing the relatedness of terms within a large corpus of text (corpus-based approach) [47]. The major techniques classified under corpus-based approach are Pointwise Mutual Information (PMI) [8] and Latent Semantic Analysis (LSA) [21], though PMI outperform LSA on mining the web for synonyms [68]. A group of Google researchers proposed two efficient models which can discover semantic word similarities [48]. The two novel models are the following:

1. Continuous Bag-of-Words model
2. Continuous Skip-gram model

These models aim to use large scale Nural Netwrok to learn word vectors. The two models have restrictions that make them not suitable in our case. The first restriction is that both models require words and context in which those words are used. In their experiments, the authors built vectors of at least 50 words around the given word (words before and after the given word from the text in which that word is used). One more restriction is that they allow only single token words to be processed (no phrases). In our case, the two models are not applicable since the searches conducted by the users usually contains a single word with no context or other words surrounding it. Also, we care about phrases since they also widely used in our search engine, for example "Java Developer" should be considered as one phrase when we discover the semantically related words. It is also mentioned in that paper that it takes less than a day to learn high quality vectors from 1.6 billion words. In our experiment, we discovered the high quality semantic relationships using a data set of 1.6 billion search

logs (search keywords used to search for jobs in Careerbuilder.com), PGMHD completed that task in 45 minutes.

Motivation

We would like to create a language-independent algorithm for modeling semantic relationships between search phrases that provides output in a human-understandable format. It is important that the person searching can be assisted by an augmented query without us creating a black-box system in which that person is unable to understand and adjust the query augmentation. CareerBuilder[1] operates job boards in many countries and receives tens of millions of search queries every day. Given the tremendous volume of search data in our logs, we would like to discover the latent semantic relationships between search terms and phrases for different region-specific websites using a novel technique that avoids the need to use natural language processing (NLP). We wish to avoid NLP in order to make it possible to apply the same technique to different websites supporting many languages without having to change the algorithms or the libraries per-language.

It is tempting to suggest using a synonym dictionary since the problem sounds like finding synonyms, but the problem here is more complicated than finding synonyms since the search terms or phrases on our site are often job titles, skills, and company names which are not, in most cases, regular words from any dictionary. For example if a user searched for "*java developer*", we would not find any synonyms for this phrase in a dictionary. Another user may search for "hadoop" which is also not a word that would be found in a typical English dictionary.

[1] http://www.careerbuilder.com/

Table 3.1: Input data to PGMHD over hadoop

UserID	Classification	Search Terms
user1	Java Developer	Java, Java Developer, C#, Software Engineer
user2	Nurse	RN, Rigistered Nurse, Health Care
user3	.NET Developer	C#, ASP, VB, Software Engineer, SE
user4	Java Developer	Java, JEE, Struts, Software Engineer, SE
user5	Health Care	Health Care Rep, HealthCare

Probabilistic Semantic Similarity Scoring using PGMHD

We applied the proposed PGMHD model to discover the semantically related search terms by measuring probabilistic-based semantic similarity between those search terms. Given the search logs for all the users and the users' classifications as shown in Table 3.1, PGMHD can represent this kind of data by placing the classes of the users as root nodes and placing the search terms for all the users in the second level as children nodes. Then, an edge will be formed linking each search term back to the class of the user who searched for it. The frequency of each search term (how many users search for it) will be stored in the node of that term, while the frequency of a specific search term searched for by users of a specific class (how many users belonging to that class searched for the given term) will be stored in the edge between the class and the term. The frequency of the root node is the summation of the frequencies on the edges that connect that root node with its children (Figure 3.16).

Distributed PGMHD

In order to process 1600 million search logs (each search log contains one or more keywords used by a user to search for jobs on careerbuilder.com) provided by Careerbuilder in reasonable time, we designed a distributed PGMHD using Hadoop HDFS [60], Hadoop

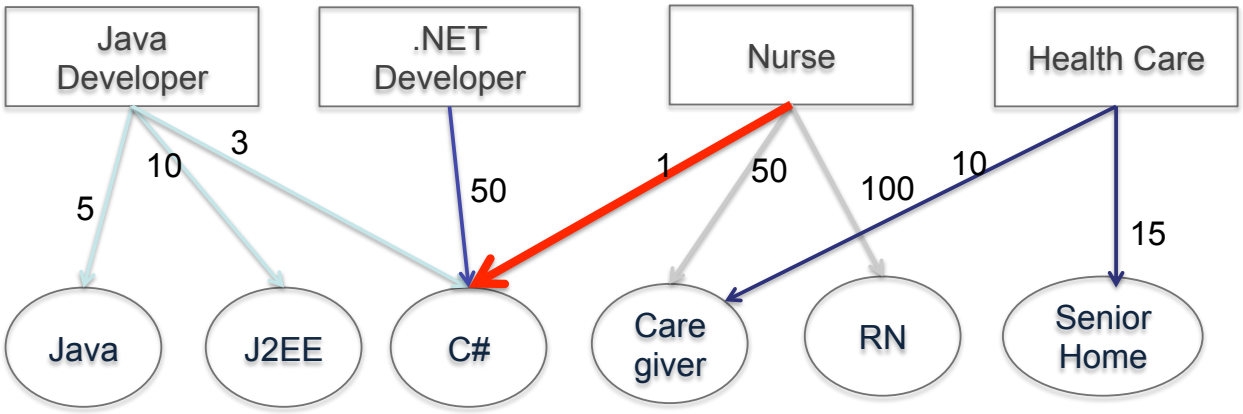


Figure 3.16: PGMHD Representing Job Search Keywords. The root nodes are job titles which each registered user has to be classified to one of them. On the second level of the graph, each node represents a search term extracted from the search logs. An edge from job title node to a search term node represents a search conducted by a user with that job title using that search term. The number on each edge that connects job title with an edge represents how many distinct users from that job title searched for that search term.

Map/Reduce [20] and Hive [66]. The design of distributed PGMHD is shown in Figure 3.17. Basically we use Hive to store the intermediate data while we are buidling and training PGMHD. Once it is trained we can then run our inquires to get an ordered list of the semantically related keywords for a specific term(s).

Experiment Setup and Results

The experiment performing latent semantic discovery among search terms using PGMHD was run on a Hadoop cluster with 69 data nodes, each having a 2.6 GHz AMD Opteron Processor with 12 to 32 cores and 32 to 128 GB RAM. Table 3.2 shows sample results of 10 terms with their top 5 related terms discovered by PGMHD. To evaluate the model's accuracy, we sent the results to data analysts at CareerBuilder who reviewed 530 random pairs of discovered related search terms and returned the list with their feedback about whether each pair of discovered related terms was "related" or "unrelated". We then calculated the accuracy

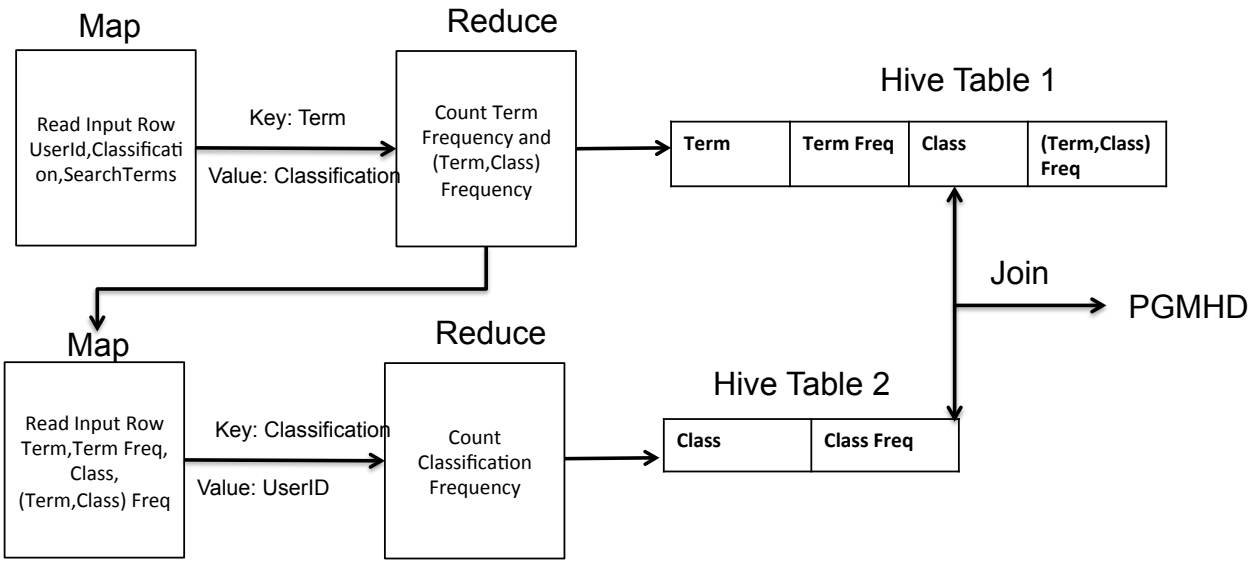


Figure 3.17: PGMHD Over Hadoop

(precision) of the model based upon the ratio of the number of related results to the total number of results. The results show the accuracy of the discovered semantic relationships among search terms using the PGMHD model to be 0.80 with 425 related pairs.

Table 3.2: PGMHD results for latent semantic discovery

Term	Related Terms
hadoop	big data, hadoop developer, obiee, java, python
registered nurse	rn registered nurse, rn, registered nurse manager, nurse, nursing, director of nursing
data mining	machine learning, data scientist, analytics, business intellegence, statistical analyst
solr	lucene, hadoop, java
Software Engineer	software developer, programmer, .net developer, web developer, software
big data	nosql, data science, machine learning, hadoop, teradata
Realtor	realtor assistant, real estate, real estate sales, sales, real estate agent
Data Scientist	machine learning, data analyst, data mining, analytics, big data
Plumbing	plumber, plumbing apprentice, plumbing maintenance, plumbing sales, maintenance
Agile	scrum, project manager, agile coach, pmiacp, scrum master

Discover Semantic Ambiguity of a Keyword

The semantic ambiguity of a keyword can be defined different meanings of the same keyword in different context [37, 30]. The techniques mentioned in the literature focus on utilization of ontologies and dictionaries like Wordnet as described in [37, 30]. Those solutions are not applicable when the keywords are from a domain like job search. In the job search domain the used keywords are either job titles, skills, company names, etc. which are not the regular English keywords. For example, Java means programming language as well as coffee, but no English dictionary will provide those two meanings.

PGMHD is applied successfully to discover the semantic ambiguity of a keyword. About 1.6 billion search logs were used in this experiment. The search keywords extracted from these 1.6 billion logs used to train PGMHD which then are used to calculate the normalized Pointwise Mutual Information (PMI) [8] score for each term with all of its parents. We determine the ambiguous keyword by applying the following technique:

Let:

- $C := \{C_1, \ldots, C_n\}$ be the set of different job classes of jobs (Java developer, Nurse, Accountant, ... etc);
- $\mathcal{T} = \{t_1, \ldots, t_N\}$ be the set of different search terms used by the users when they conducted searches (N is the number of different terms); and
- $f(C_j, t)$ be is the number times (frequency) a user from class $C_j \in C$ searched for the keyword $t \in \mathcal{T}$.
  - We will only consider the frequencies with at least 100 distinct searches, i.e.,

$$f(c, t) \geq 100.$$

Then, define

- $O(c)$: the number of times a user from class c searched for a keyword i.e.:

$$O(c) := \sum_{t \in T} f(c,t) \qquad c \in C;$$

- $T(t)$: the number of times the keyword t_j is searched, i.e.:

$$T(t) := \sum_{c \in C} f(c,t) \qquad t \in \mathcal{T};$$

- T :the total number of keyword searches, i.e.:

$$T := \sum_{c,t} f(c,t) = \sum_{c \in C} O(c) = \sum_{t \in \mathcal{T}} T(t).$$

We estimate the PMI

$$\frac{\mathbb{P}(C=c, T=t)}{\mathbb{P}(C=c)\mathbb{P}(T=t)} = \frac{\mathbb{P}(C=c|T=t)}{\mathbb{P}(C=c)}$$

Which represents the probability the keyword t is searched by a user from class c with the following estimate

$$\text{pmi}(c,t) := \log \frac{f(c,t)}{T(t)} \frac{T}{O(c)} \qquad c \in C, t \in \mathcal{T}.$$

The normalized version of the original pmi [8] estimate given by

$$\begin{aligned} \tilde{p}(c,t): &= \frac{\text{pmi}(c,t)}{-\log \frac{f(c,t)}{T}} = \frac{\log T + \log f(c,t) - \log\left[O(c)T(t)\right]}{\log T - \log f(c,t)} \\ &= -1 + \frac{2\log T - \log O(c) - \log T(t)}{\log T - \log f(c,t)} \in [-1,1] \qquad c \in \mathcal{C}, t \in \mathcal{T}. \end{aligned}$$

Ambiguity score

For every search keyword $t \in \mathcal{T}$, we define the following *ambiguity score* $A_\alpha(t)$ as

$$A(t) := |\{i : \tilde{p}(c,t) > 0\}|,$$

we say that a search keyword t_j is a *candidate* to be ambiguous if $A_j(\alpha) > 1$. Then, we can define a set of candidate ambiguous terms CA as

$$CA = \{t_j : A_j(\alpha) > 1, j = 1, \ldots, N\}.$$

Table 3.3 shows sample results of the discovered semantically ambiguous terms.

Table 3.3: PGMHD results for semantic ambiguity discovery. The first column shows the keyword, while the second column shows the related keywords of each possible meaning separated by horizontal line

term	**related terms of meaning**
architect	enterprise architect, java architect, data architect, telecommute, oracle, java, .net
	architectural designer, architectural drafter, autocad, autocad drafter, designer, drafter, cad, engineer
account	bookkeeper, accountant, analyst, finance
	sales executive, account executive, insurance, account manager, outside sales, medical sales, manager, sales
cna	forklift, forklift operator, sales, facilities maintenance, hvac
	cna certified nursing assistant, certified nursing assistant, nursing assistant, call center, patient care technician, dental assistant, customer service rep, cashier, phlebotomist, medical assistant,
designer	design, print, animation, artist, illustrator, creative, graphic artist, graphic, photoshop, video
	graphic, web designer, design, web design, graphic design, graphic designer
	design, drafter, cad designer, draftsman, autocad, mechanical designer, proe, drafter drafting designer autocad, structural designer, revit
writer	copywriter, communications manager, communications public relations, architecture, communications, public relations, marketing communications, social media, consultant
	editor, writer editor, writing, copywriter, technical writer, editorial, reporter, communications, proposal, proofreader

3.7 Conclusions and Future Work

Probabilistic graphical models are very important in many modern applications such as data mining and data analytics. The major issue with existing probabilistic graphical models is their lack of scalability to handle large data sets, making this a very important area for research given the tremendous modern focus on big data due to the number of data points produced by modern computers systems and sensors. PGMHD is a probabilistic graphical model that attempts to solve the scalability problems in existing models in scenarios where massive hierarchical data are present. PGMHD is designed to fit hierarchical data sets of any size , regardless of the domain to which the data belongs. PGMHD can represent the hierarchical data with any number of levels, it can handle multi-label classification, and it is suitable for progressive learning since it is considered as lazy classifier. In this paper we present three experiments from different domains: one being the automated tagging of high-throughput mass spectrometry data in bioinformatics, the other being latent semantic discovery using search logs from the largest job board in the U.S, and the last one being semantically ambiguous keywords discovery. The three use cases in which we tested PGMHD show that this model is robust and can scale from a few thousand entries to billions of entries, and can also run on a single computer (for smaller data sets), as well as in a parallelized fashion on a large cluster of servers (69 were used in our experiment). PGMHD is used in production at Careerbuilder.com for the semantically related keywords discovery and discover semantic ambiguous keywords. The work on the discovery of semantic ambiguous keywords will continue. We plan to publish a separate paper about it. Also, we plan to compare machine learning algorithms implemented in Apache Spark with PGMHD.

Chapter 4

GELATO and SAGE: An Integrated Framework for MS Annotation

4.1 Abstract

Several algorithms and tools have been developed to (semi)automate the process of glycan identification by interpreting Mass Spectrometric data. However, each has limitations when annotating MS^n data with hundreds or thousands of MS spectra using uncurated public databases. Moreover, the existing tools can handle only MS/MS but not entire MS^n data at once. In this paper, we propose a novel software package to automate the MS annotation. The proposed software consists of two major components: (1) The Glycomic Elucidation and Annotation Tool (GELATO) a free, semi-automated MS^n data interpreter which was designed and implemented at the Complex Carbohydrate Research Center (CCRC). GELATO combines functionality from existing open source projects, namely, GlycoWorkbench (GWB) and GlycomeDB, with improvements to the GlycoWorkbench annotation algorithm and extensions to automate the interpretation process. (2) The Smart Anotation Enhancement Graph (SAGE) is a machine learning model that learns the annotation behavior of the user

to predict the best annotations among those generated by GELATO for future MS annotation runs.

4.2 Introduction

Along with nucleic acids, proteins, and lipids, complex carbohydrates, also known as glycans, comprise the four major classes of macromolecules fundamental to all living systems [70]. Until recently, relatively little attention has been paid to studying glycans despite the major roles they play in diverse biological processes [22]. Almost all cells are coated with a dense layer of glycans and virtually all cellular interactions take place in the context of this layer . Most proteins that are produced by eukaryotic cells for export or insertion into the cell membrane are glycosylated and proper glycosylation is often critical for their biological functions [6, 10]. Due to their complex structure, the potential information content encoded by glycans attached to proteins exceeds that of any other post-translational modification [56]. The importance of the emerging field of glycobiology is warranted by the accumulated evidence for the role of glycans in cell growth and metastasis, cell-cell communication, and microbial pathogenesis. However, glycans are more diverse than nucleic acids and proteins [29, 70], mainly due to their branched structures, which are not encoded by a molecular template.

Full and comprehensive characterization of the glycans on glycoproteins has become an essential element for drug development, quality control, and basic biomedical research. Glycan identification is much more difficult than protein identification, and de novo glycan sequencing is a proven NP-hard problem [64]. Identification of peptides using tandem mass spectrometry (MS^n) [1] is facilitated by their linear structures and the availability of reliable peptide sequence databases. MS^n of a peptide generates a relatively complete series of high-intensity fragment ions with mass differences that correspond to specific amino

acids, providing clear-cut information regarding the peptide's amino acid sequence. However, complete ion series are rarely observed in the MS^n of glycans and the branched nature of these molecules further complicates sequence determination. In addition, the monosaccharide building blocks of glycans comprise several isomeric sets (e.g., the set of all hexoses) whose members can be rarely distinguished by MS.

The existing glycan structure databases are limited, minimally curated, and frequently polluted with erroneous sequences or structures that are not found in natural sources or are irrelevant to the organism of study. Current glycomics technology thus relies heavily on the manual interpretation of mass spectrometry datasets. Furthermore, as instrumentation improves, dataset size increases (e.g., 2000 or more mass spectra per biological sample), thus demanding even more time for manual interpretation and reducing the number of samples that can be analyzed. This bottleneck is a major impediment blocking the application of glycoanalysis to a broad range of important biomedical investigations. By analyzing to the existing tools of MS annotation, we found that each tool has limitations in one or more aspects. GlycoMod [17] which is a web based tool that offers free MS annotation which has many limitations. It supports limited derivitisation and labeling adducts, with no support to neutral exchange. It is made for MS profile annotation, but not MS^n annotation. It annotates using compositions rather than structures. GlycoPeakfinder [46] is another web based tool that annotates MS profile and MS^n spectra. However, it only allows annotation of a single spectra at a time, and it annotates with compositions rather than structures. GlycoWorkbench (GWB) [13] offers free glycan annotation of MS^n spectra using user defined structures or structures from a database. Its main limitations are that spectra need to be uploaded one at a time, slowing down the data processing steps. Furthermore, undermethylation of glycan structures [14] is not considered in the annotation process. The commercial tool SimGlycan® offers glycan annotation of MS^n spectra by using glycan structures from an integrated databases based on KEGG [40]. It allows high throughput by uploading and

annotating entire MS^n runs, but also does not consider undermethylation of glycans (Table 4.1 compare different MS annotation tools). In this paper we present a software package consisting of two components assisting in the interpretation of MS^n data: GELATO a freely available algorithm for the annotation of MS^n spectra of glycans and SAGE a machine learning model for refining GELATO annotations with trained expert knowledge.

4.3 Methods

Glycomic Elucidation and Annotation Tool (GELATO) is a freely available, semi-automated MS^n interpreter for glycans which was designed and implemented at the Complex Carbohydrate Research Center (CCRC). GELATO extends functionality from existing open source projects, namely, GlycoWorkbench [13] and GlycomeDB [58].

The extensions implemented in GELATO which are not part of the GWB annotation algorithm are the following:

1. Uploading an intact MS^n run at once rather than each spectrum separately.
2. Creation of new adducts. In case the user would like to annotate by considering an adduct not predefined in our list, he/she can define a new adduct by providing the required information including name, charge, and mass.
3. Supporting of multiple charged adducts.
4. Creation of new ions for the netral ion exchange.
5. Supporting loss of small molecules.
6. Undermethylation.
7. Fragmentaiton settings per MS level or per activation method.

8. Supporting different accuracy settings for MS^1 and MS^n. This allows providing tolerance value for MS^1 differ than the one to be used with MS^n. The importance of this is that in MS analysis the tolerance value for MS^1 is usually bigger than the one for MS^n in order to improve the accuracy of the annotation.

A set of human curated databases for different types of glycans (e.g, N-glycans, O-glycans and Glycosphintolipids) [23], which are generated from the Glycan Ontology, GlycO [65] are used as the source of glycan structures for the annotations, increasing confidence in the results produced by GELATO. Figure 4.1 and Figure 4.2 show the annotations produced by GELATO for MS^1 and MS^n, respectively. Due to space limitation, the annotation workflow used by GELATO is provided in the supplementary document. The GELATO annotation process starts by setting parameters that specify the types of fragmentation events the user deems likely to generate the spectra being processed, the adducts used in the experiment, neutral ion exchange and fragmentation effects that may lead to the loss of molecular parts (e.g. Water for non-deriatices glycans, Methl for permethlylated glycans or the loss of charged monosaccharides during ionization). Glycan structures are then retrieved from the chosen database one-by-one. Each glycan is checked to determine whether its calculated mass corresponds, within a specified tolerance, to the m/z of any precursor ion detected in MS^1. If such a match is found, fragmentation occurs *in silico* using the given user specified settings. Calculated m/z values and structure of each theoretical fragment ion is saved. The m/z values of the pre-calculated and stored fragment ions of this glycan are compared with the m/z values observed in the spectrum generated by selecting the precursor ion. Observed fragment ions with m/z values matching to ions of glycan fragments are annotated with these fragments. The resulting annotations are immediately serialized to a data file as they are assigned, making it possible for GELATO to run on a desktop computer or laptop with limited memory space and CPU to handle MS data files containing hundreds of thousands of spectra. For MS^n spectra this process is repeated recursively for all sub spectra by choosing

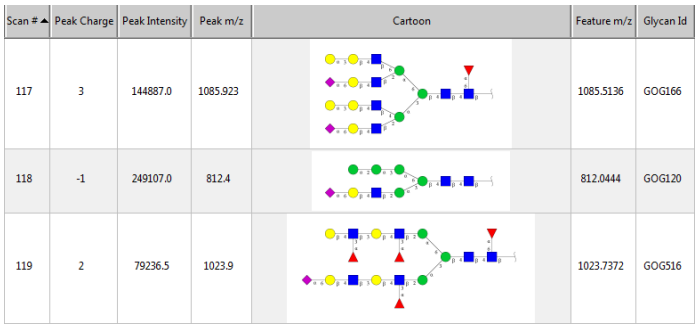


Figure 4.1: MS^1 annotations using GELATO

the fragment annotations of MS^n precursors for further fragmentation and annotation of the sub spectra. GELATO generates and records all of the possible annotations based on these criteria for each processed spectrum without applying human expert knowledge or result filtering. The annotations are then ranked using two complementary scores. The first score, $score^c$, corresponds to the number of fragment annotations for glycan G^x divided by the total number of ions in scan S^y, as shown in Equation 4.1 in Figure 4.3. The second score is based on relative intensity of annotated ions compared to the total intensity of all ions in the spectra. The structurally relevant ions in a given peak list (spectrum) usually exhibit high intensity. However, the $score^c$ incorporates all annotated peaks regardless of their intensity, which can lead to less meaningful rankings if a large number of noise peaks are annotated because they happen to have m/z values that match the simulated spectrum of a particular glycan fragment. This issue is addressed by calculating the relative intensity score ($score^i$), which corresponds to the fraction of the total observed ion intensity in a fragment-ion spectrum that is derived from annotated peaks. Given a glycan G^x and a scan S^y. $score^i$ is calculated using Equation 4.2 in Figure 4.3. However, $score^i$ may also result in misleading ranking if only one or few high intensity peaks are annotated. Therefore, both scores are provided to assist in user evaluation of the annotation results.

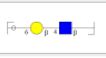
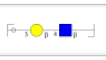
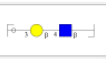
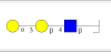
Peak Intensity	Peak m/z	Cartoon	Feature m/z	Glycan Id
13.1097	398.3249		398.1693	GOG166
5.2044	445.3975		445.1952	GOG166
13.2528	472.3444		472.2061	GOG166
13.2528	472.3444		472.2061	GOG166
13.2528	472.3444		472.2061	GOG166
8.7515	474.3912		474.2217	GOG166
2.8145	690.4586		690.3215	GOG166
2.8145	690.4586		690.3215	GOG166

Figure 4.2: MS^n annotations using GELATO

$$score_c(G_x|S_y) = \frac{num(AP_xy)}{num(P_y)} \tag{4.1}$$

$$score_i(G_x|S_y) = \frac{\sum I(AP_xy)}{\sum I(P_y)} \tag{4.2}$$

Figure 4.3: Scores in GELATO. $num(AP_xy)$ is the number of annotated peaks in S_y by fragments generated by G_x, $num(P_y)$ is the total number of peaks in S_y. $I(AP_xy)$ intensity of annotated peak in S_y by fragments in G_x, and $I(P_y)$ is the intensity of a peak in S_y

The annotation generation and ranking process provide all theoretically possible annotations, many of which may be meaningless. The user must therefore review the annotations and eliminate those that are judged to be incorrect, which can take considerable time and effort, especially when high throughput data are processed. In order to speed up the annotation of MS data, we thus implemented a machine learning model that identifies meaningful annotations based on human review patterns of previously processed datasets. This algorithm was implemented as a customization of the Probabilistic Graphical Model for Massive Hierarchical Data (PGMHD) [2]. The aim of the resulting tool, which we call Smart Annotation Enhancement Graph (SAGE), is to learn the annotation behavior of a user or group of users and apply this annotation behavior in future annotations. The tool builds a probabilistic graph model that represents glycans and glycan fragments previously selected as meaningful annotations by the user and utilizes this graph to calculate the probability that the user would accept or reject a given annotation of the new dataset. SAGE can be used in two contexts, as an annotation tool, where it actually generates, scores and selects annotations, or as post-filtering tool, where it analyzes previously annotated spectra (e.g, processed by GELATO) to reject annotations that are unlikely to be accepted by the user.

Training SAGE is straightforward and can be accomplished either in a single session or over many sessions. For example, spectra that have been annotated using GELATO are reviewed by the user who selects a subset of the provided annotations that he/she judges to be correct. The selected annotations are processed by SAGE to either build a new probabilistic graph model or integrate the new annotations into an existing model. The proposed learning algorithm for SAGE, shown in Algorithm 2, is designed to facilitate progressive learning, which has the following advantages:

1. Data required for training the model can be generated, evaluated by the user and processed in stages and at different times.

2. New training data is easily incorporated to extend the model without reprocessing data used in previous training sessions.

3. Training can be distributed over many sessions, eliminating the need for a single, prolonged session, which might fail and have to be repeated.

4. Recursive learning is possible, allowing the model itself to generate new training data by processing new MS data sets, provided that the new annotations are judged to be accurate by the user.

SAGE approaches MS annotation as a multi-label classification problem. Complete glycan structures that can annotate the spectra are the classes (into which observed spectra are assigned) while the fragments used to annotate the observed MS peaks are treated as features. Figure 4.4 illustrates the representation of MS data in SAGE. Each root node is labeled with a glycan structure and a specific m/z value, indicating that the user approved that structure during the training phase to annotate a precursor ion observed at that m/z. The nodes in the lower levels represent the fragments approved to annotate MS^2, MS^3, MS^n peaks. Edges are allowed between nodes at level i to nodes at level $i+1$ if and only if the glycan or the fragment at level i decomposes to generate the fragment ion at level $i+1$. Numeric edge labels represent the number of times the parent node (the source of the edge) appears in the training data in association with the child node (the destination of the edge).

To use SAGE, the model associates the precursor ion of the scan being processed with a set of root nodes (classes) and associates m/z values from the scan (peak list) with features in the trained model. The probabilistic classification score, $P(G^x \mid f^1, f^2, f^3, f^n)$ of each glycan (*root node*) is calculated given observation of those features in the scan. In order to optimize the search space, only those glycans forming quasi-molecular ions with m/z values that are within a specified tolerance from the precursor m/z for the given scan are considered, Figure 4.6 shows the annotation workflow for SAGE. The probabilistic score is

```
Data: Annotated MS^n Spectra Using GELATO
Result: SAGE Instance
begin
    currentMSLevel = 0
    while currentMSLevel < maxMSLevel − 1 do
        foreach annotatedPrecursor ∈ currentMSLevel do
            if annotatedPrecursor.annotation exists in SAGE.currentLevelNodes
            then
                get sageNode where sageNode.annotation =
                annotatedPrecursor.annotation
                sageNode.frequency+ = 1
            else
                sageNode = newnode
                sageNode.frequency = 1
            end
            childrenLevel = currentMSLevel + 1
            foreach annotatedChildPeak ∈ annotatedPrecursor.peakList do
                foreach sageChildNode ∈ sageNode.children do
                    if annotatedChildPeak.annotation = sageChildNode.annotation
                    then
                        edge = edge(sagedNode, sageChildNode) edge.frequency+ = 1
                    else
                        if childNode ∈ sage.childrenLevelNodes then
                            edge = createNewEdge(sageNode, sageChildNode)
                            edge.frequency = 1
                        else
                            sageChildNode = newNode sageChildNode.annotation =
                            annotatedChildPeak.annotation
                            sageChildNode.frequency = 1
                            edge = createNewEdge(sageNode, sageChildNode)
                            edge.frequency = 1
                        end
                    end
                end
            end
        end
        currentMSLevel = currentMSLevel + 1
    end
end
```

Algorithm 2: Learning Algorithm for SAGE. *currentMSLevel* represents the current MS level in the MS data we are processing, we start with level 0 which is related to MS level 1. *maxMSLevel* is the highest level in the given MS data.

calculated as described in [2]. For example, to use SAGE instance in Figure 4.4, assume we have a spectra with only the fragments F_1, F_3, F_7 and we would like to know which glycan structure in the root level is the best annotation. We can calculate the probabilistic score $P(G_1|F_1, F_3, F_7)$ and $P(G_2|F_1, F_3, F_7)$ as described in [2].

$$P(G_1|F_1, F_3, F_7) = P(G_1|F_1, F_3) \cdot P(F_3|F_7) = P(G_1|F_1) \cdot P(G1|F_3) * cdot(F_3|F_7)$$

$$= 50/50 \cdot 20/60 \cdot 10/25 = 0.13$$

$$P(G_2|F_1, F_3, F_7) = P(G_2|F_1, F_3) \cdot P(F_3|F_7) = P(G_2|F_1) \cdot P(G2|F_3) \cdot (F_3|F_7)$$

$$= 0.1 \cdot 40/60 \cdot 10/25 = 0.02$$

Since there is no edge between G_2 and F_1 while it is a valid and possible fragment to G_2, we used m-estimate as described in [2] to resolve the zero probability problem. If any fragment is not possible to be generated by a glycan it will not be included due to the selectivity of the model.

4.4 Discussion

The proposed system combines the semi-automated MS annotation tool GELATO with the machine learning model SAGE to automate the process of MS annotation, this integration is shown in figure 4.5. In (A) the training of SAGE depends on the annotations generated by GELATO. Those annotations are reported to the user who can select a subset of the generated annotations as the desired annotations. These annotations are used to train SAGE. In (B) the trained SAGE is applied to either annotate a new spectra or to improve the annotations generated by GELATO for a given spectra. This post-filtering process uses

Table 4.1: Features of the existing MS annotation tools (part I). Annotates MS^n, reflects the ability of the tools to annotate MS^n spectra, not just MS profile. Annotates with structures, means the tool uses database of glycans for annotation. Can find novel structures, means it can annotate using new glycan structures it never used before for annotation. Scores, define what type of scores the tools support. Statistical means scores calculated using statistical methods (summation, counting,etc.), while probabilistic means scores calculated based on probability distribution.

Tool	Annotates MS^n	Annotates with structures	Can find novel structures	Scores
GELATO	Yes	Yes	Yes	Statistical
SAGE	Yes	Yes	No	Probabilistic
SAGE and GELATO	Yes	Yes	Yes	Statistical and Probabilistic
GWB	Yes	Yes	Yes	Statistical
GlycoMod	No	Yes	Yes	Statistical
SimGlycan	Yes	Yes	Yes	Statistical

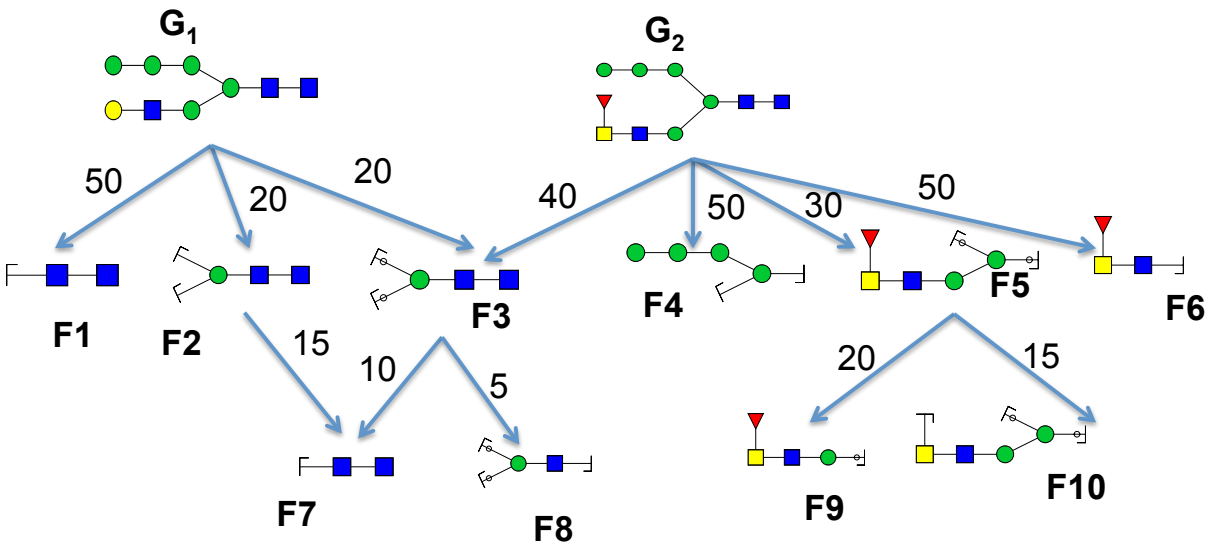


Figure 4.4: SAGE representing the MS data up to MS^3. The root nodes are glycans used to annotated precursors in the training data, while the nodes at lower levels represent fragments (F_i is the fragment Id) used to annotate the peaks in different MS levels. The edges represent the co-occurrence between the two nodes it connect while the number on the edge represent the frequency of that co-occurrence.

Table 4.2: Features of the existing MS annotation tools (part II). Handle under-methylation, reflects the ability of the tools to consider under-methylation during the annotation process. Handle natural loss, reflects the ability to consider the natural loss in the annotation process. Average annotation time, reflects in general how long the annotation process require to be done. And availability, is either the tool is freely available or it is a commercial one.

Tool	Handle Under-methylation	Handle Natural Loss	Uses human expert knowledge	Avg Annotation Time	Availability
GELATO	Yes	Yes	No	Minutes	Free
SAGE	No	No	Yes	Seconds	Free
SAGE and GELATO	Yes	Yes	Yes	Minutes	Free
GWB	No	No	No	Minutes	Free
GlycoMod	No	No	No	Minutes	Free
SimGlycan	No	No	No	Hours	Commercial

the knowledge learned by SAGE to eliminate the annotations which most likely will be eliminated by the user.

SAGE calculates the probabilistic score for each possible annotation (scan-glycan pairing), which reflects the likelihood that this annotation would be accepted by the user given the knowledge previously learned by SAGE about this user (or user group). The user can instruct SAGE to report the K top-scoring annotations. Otherwise, SAGE will report all the possible annotations ranked by their probabilistic score. This integrated framework is able to handle the shortcoming of the existing tools as shown in Table 4.1 and Table 4.2.

Since human knowledge is subjective, so what seems correct to a person does not mean it is correct for another person, at this point SAGE not intended building a global knowledge-base which can be applied everywhere by everyone. Each instance of SAGE represents a user's knowledge which he/she would like to use in the future MS annotations. One other important aspect to have an accurate annotations and efficient usage of SAGE is to train

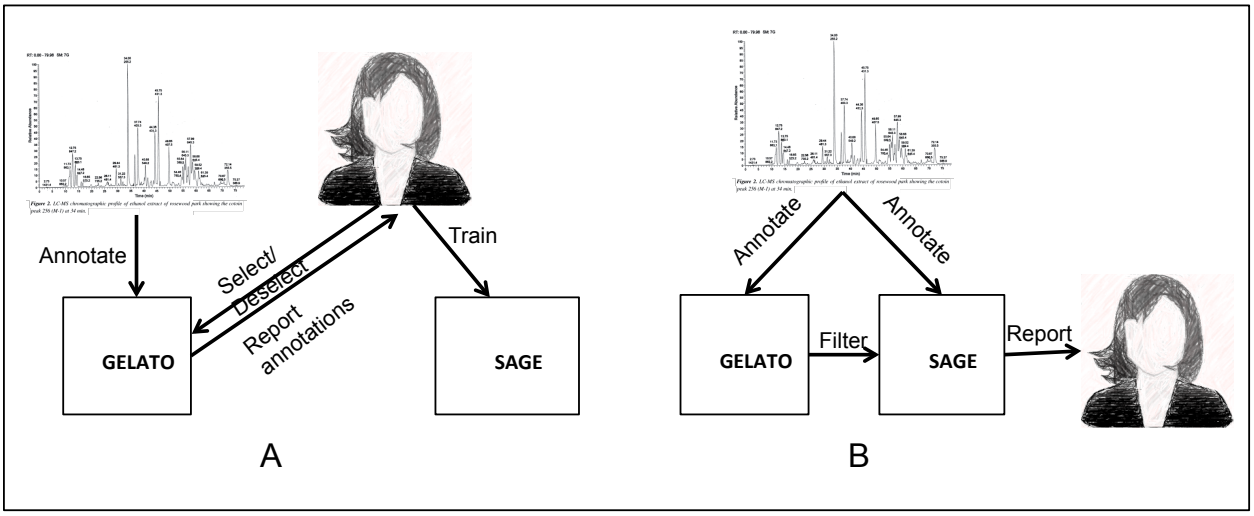


Figure 4.5: The integration between SAGE and GELATO. (A) GELATO annotates a given spectra, then a user select subset of those annotations and provide this final list of approved annotations to train SAGE. (B) the trained SAGE can be used either to annotate the given spectra or to filter out the annotations calculated by GELATO which most likely will not be selected by the user.

a new instance of SAGE to annotate MS experiments of different species since The glycans present in human samples may not be possible in other species and vice versa.

4.5 Experiment and Results

To test the proposed framework, we used GELATO to annotate 10 MS datasets generated during the glycoanalysis of stem cells at the CCRC. Annotations generated by GELATO were compared with annotations generated by SimGlycan, a commercial software tool for MS annotation. For each spectrum tested, GELATO generated all the annotations generated by SimGlycan. However, GELATO was able to provide annotations for some scans that SimGlycan could not annotate due to two reasons. The first one is the limitation of SimGlycan to annotate a scan which appears in the mzXML file with a charge state -1, which

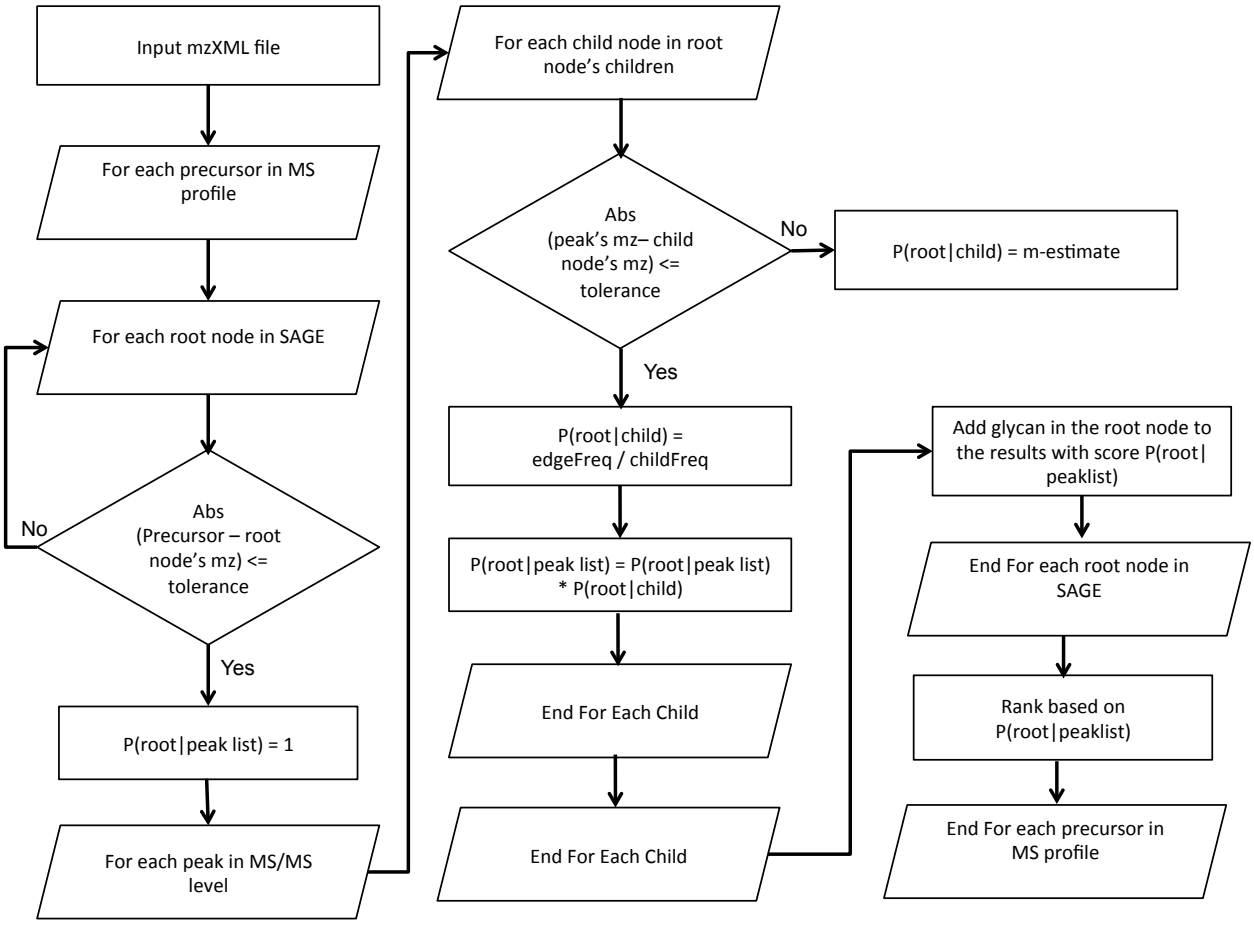
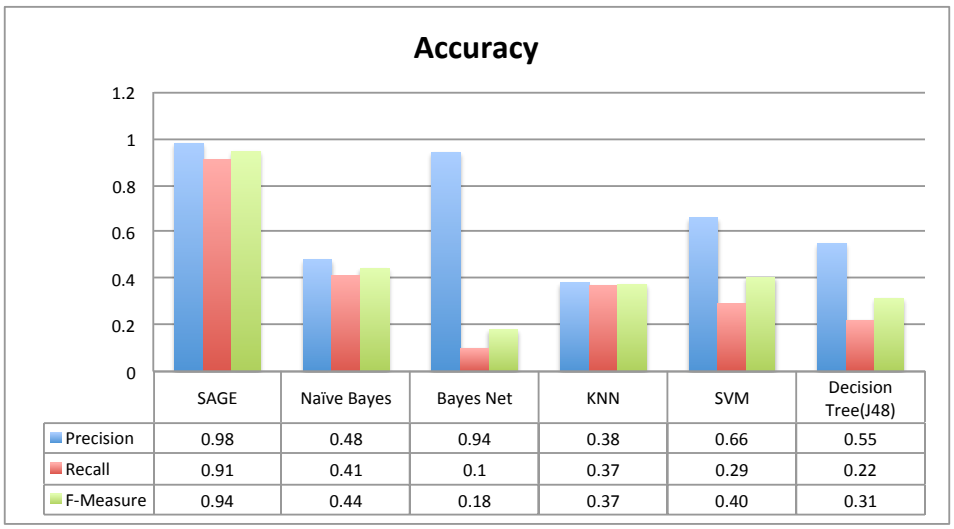


Figure 4.6: The annotation workflow of SAGE. For more information about m-estimate, childFreq, edgeFreq and probabilistic based score please see [2]

indicates that the charge state could not be determined by the instrument. GELATO with its feature of handling multi-charge state can annotate those scans with charge state assigned as -1, by trying different charge states starting at 1 until the maximum charge given by the user is reached. The second reason is the incomplete database which SimGlycan uses. The integrated annotation tool was tested in a second experiment, where GELATO was used to annotate MS data from 10 glycoanalysis experiments conducted at the CCRC on pancreatic cancer samples [54]. An MS expert reviewed the annotations generated by GELATO and selected a subset that she judged as correct. Nine of the ten curated annotations were used to train SAGE, while the tenth was used for testing. The trained SAGE was used to generate *denovo* annotations of the tenth MS data set and these annotations were compared to those generated by GELATO and approved by the expert for the same dataset. This process was repeated with different test sets and training sets 10 times, then we calculated the average accuracy. By these criteria, the annotations generated by SAGE were 98% accurate in average with 91% coverage (it predicted 91% of the manually approved annotations of the test spectra). Since SAGE functions as a multi-label classifier and the MS annotation is multi-label classification problem, we compared SAGE with the top classifiers in machine learning. We used Mulan [67] which is an extension to the well-known machine learning library Weka to handle multi-label classification. Figure 4.7 shows that the SAGE implementation of PGMHD outperforms all the well-known classifiers in this challenging task. We plan to integrate this annotation framework in the GRITS-Toolbox (http://www.grits-toolbox.org), a standalone application for the interpretation and annotation of glycomics MS data. Given that the GRITS-Toolbox is designed to run on a single workstation (desktop or laptop) with limited memory, control of memory usage is a critical issue. In this context, we compared the memory usage of SAGE compared to the other classifiers. Figure 4.8 shows that the SAGE implementation of PGMHD used less memory (54 megabytes in average) than the other machine learning models to represent the training dataset. Another critical aspect of



	SAGE	Naïve Bayes	Bayes Net	KNN	SVM	Decision Tree(J48)
Precision	0.98	0.48	0.94	0.38	0.66	0.55
Recall	0.91	0.41	0.1	0.37	0.29	0.22
F-Measure	0.94	0.44	0.18	0.37	0.40	0.31

Figure 4.7: Precision, Recall, and F-Measure of the SAGE compared to the most popular classifiers

the training of and the annotation by SAGE its time complexity. Figure 4.9 shows that, of all the tested machine learning models, SAGE required the least training time when using the same training dataset. Figure 4.10 compares the annotation time using different machine learning models. Among these, SAGE is the third fastest in this respect.

4.6 Conclusion

We designed and implemented an integrated software package for automated MS annotation, consisting of two major components, GELATO and SAGE. GELATO is a semi-automated annotation tool which is built upon the open source projects, GlycoWorkbench and GlycomeDB. SAGE is a novel machine learning model that mimics the annotation patterns of the user to determine whether annotations of new data are likely to be accepted or rejected by the user. The current implementation of this framework utilizes GELATO to generate annotations that are used to train SAGE - SAGE can then be used to either annotate MS

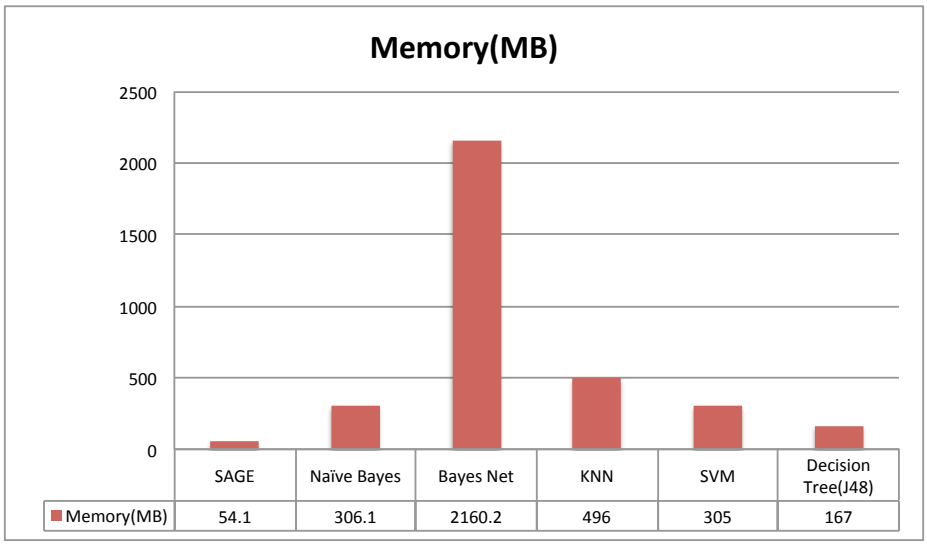


Figure 4.8: Main memory usage by SAGE compared to the other machine learning models for the training dataset

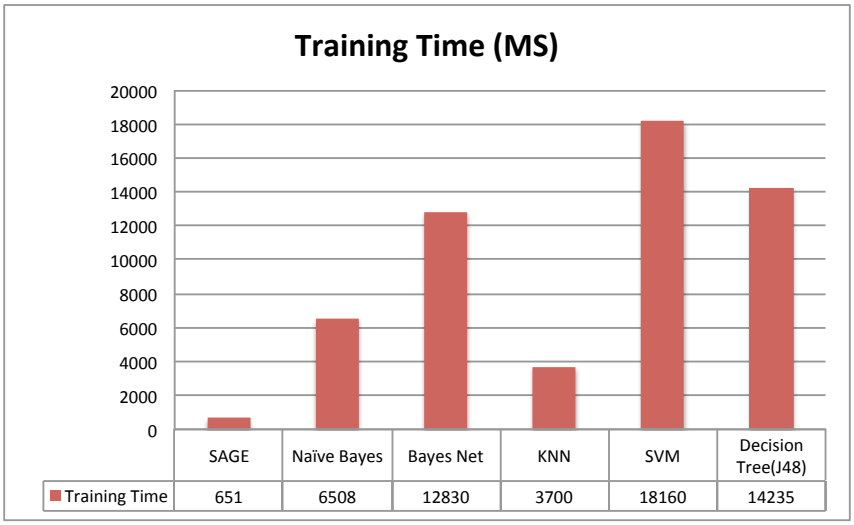


Figure 4.9: Training time for different models in millisecond. SAGE is the fastest model to learn and converge.

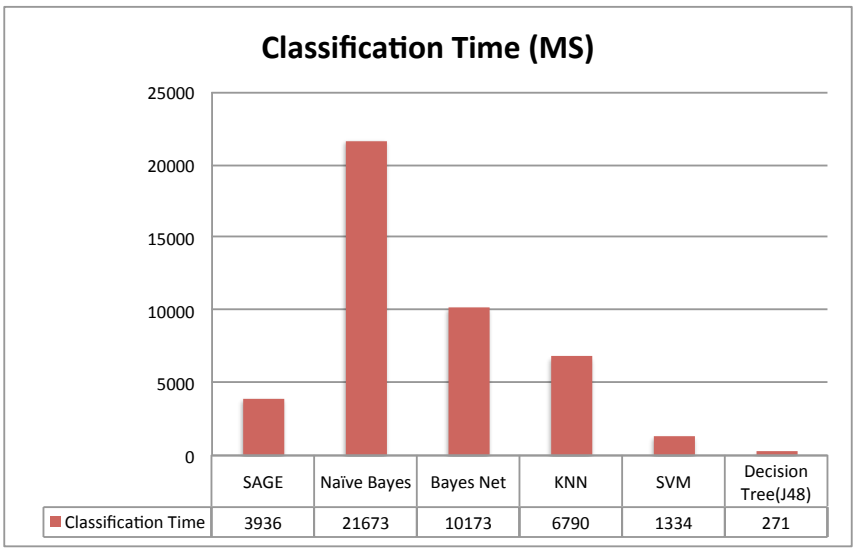


Figure 4.10: Annotation time for different models in millisecond. SAGE is the third fastest model.

spectra or improve the annotations generated by GELATO, providing a probabilistic score that can be used as the basis for rejecting incorrect annotations.

Acknowledgement

We would like to thank Mohammed Korayem from Indiana University, Camilo Ortiz, and Trey Grainger from Careerbuilder.com for their contribution to the PGMHD model which we customize to SAGE. Also, we would like to thank Kiyoko Aoki Kinoshita from Soka University for the valuable time and deep discussion with her during the design of GELATO and SAGE. Many thanks to the team of EUROCarbDB for the well designed open source libraries which we built GELATO upon. This work was supported by the National Institute of General Medical Sciences, a part of the National Institutes of Health, funding the National Center for Biomedical Glycomics (8P41GM103490).

Chapter 5

Summary

Machine learning algorithms are considered the core of data analysis and data driven computation. Most of those algorithms exhibit scalability limitations which make it difficult to utilize them with big data. Since the trend these days is to analyze huge datasets to discover new insights and hidden patterns, scaling up machine learning algorithms is very important. In this thesis, we propose a scalable probabilistic graphical model which can be considered as an extension to a well known machine learning model, Bayesian Networks. Bayesian Networks are used in many disciplines for causality modeling, classification, system reliability models, etc. The scalability issue with Bayesian Networks is caused mainly by the learning algorithms which aim to calculate the optimal Bayesian Network structure that satisfies two conditions, maximize the posterior probability and minimize the complexity of the structure. When the number of random variables is very large or have huge domains, general Bayesian Networks may become intractable. It has been proven that learning the optimal structure of Bayesian Networks is an NP-hard problem. Also, in terms of space complexity, with huge domains or sets of random variables, the CPTs in Bayesian Networks becomes huge. In the proposed model, we estimate the optimal structure of Bayesian Networks for massive hierarchical data by the following:

1. Restricting the number of parents for the posterior probability calculation.

2. Replacing the CPTs with frequencies of the co-occurrence between parents and children.

The proposed model is successfully applied in two domains, one is bioinformatics, while the other one is search logs analysis. The model successfully scales from handling thousands of data entries on a single computer to billions of data entries on a Hadoop cluster of 69 nodes. We plan to find more use cases for the proposed models, as well as compare it with the machine learning algorithms implemented in Apache Spark.

References

[1] Ruedi Aebersold and Matthias Mann. Mass spectrometry-based proteomics. *Nature*, 422(6928):198–207, 2003.

[2] Khalifeh AlJadda, Mohammed Korayem, Camilo Ortiz, Trey Grainger, John A Miller, and William S York. Pgmhd: A scalable probabilistic graphical model for massive hierarchical data problems. In *Big Data, 2014 IEEE International Conference on*, pages 55–61. IEEE, 2014.

[3] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[4] Animashree Anandkumar, Daniel Hsu, Adel Javanmard, and Sham Kakade. Learning linear bayesian networks with latent variables. In *Proceedings of The 30th International Conference on Machine Learning*, pages 249–257, 2013.

[5] Kiyoko F Aoki-Kinoshita. An introduction to bioinformatics for glycomics research. *PLoS computational biology*, 4(5):e1000075, 2008.

[6] Rolf Apweiler, Henning Hermjakob, and Nathan Sharon. On the frequency of protein glycosylation, as deduced from analysis of the swiss-prot database. *Biochimica et Biophysica Acta (BBA)-General Subjects*, 1473(1):4–8, 1999.

[7] Concha Bielza and Pedro Larrañaga. Discrete bayesian network classifiers: A survey. *ACM Computing Surveys (CSUR)*, 47(1):5, 2014.

[8] Gerlof Bouma. Normalized (pointwise) mutual information in collocation extraction. In *Proceedings of the Biennial GSCL Conference*, pages 31–40, 2009.

[9] Philippe Broët, Sylvia Richardson, and François Radvanyi. Bayesian hierarchical model for identifying changes in gene expression from microarray experiments. *Journal of Computational Biology*, 9(4):671–683, 2002.

[10] Susan A Brooks. Strategies for analysis of the glycosylation of proteins: Current status and future perspectives. *Molecular biotechnology*, 43(1):76–88, 2009.

[11] Yingyi Bu, Vinayak Borkar, Michael J Carey, Joshua Rosen, Neoklis Polyzotis, Tyson Condie, Markus Weimer, and Raghu Ramakrishnan. Scaling datalog for machine learning on big data. *arXiv preprint arXiv:1203.0160*, 2012.

[12] Alexander Budanitsky and Graeme Hirst. Semantic distance in wordnet: An experimental, application-oriented evaluation of five measures. In *Workshop on WordNet and Other Lexical Resources*, volume 2, 2001.

[13] Alessio Ceroni, Kai Maass, Hildegard Geyer, Rudolf Geyer, Anne Dell, and Stuart M Haslam. Glycoworkbench: a tool for the computer-assisted annotation of mass spectra of glycansâĂă. *Journal of proteome research*, 7(4):1650–1659, 2008.

[14] Ping-Fu Cheng, Sergei Snovida, Ming-Yi Ho, Chu-Wen Cheng, Albert M Wu, and Kay-Hooi Khoo. Increasing the depth of mass spectrometry-based glycomic coverage by

additional dimensions of sulfoglycomics and target analysis of permethylated glycans. *Analytical and bioanalytical chemistry*, 405(21):6683–6695, 2013.

[15] David Maxwell Chickering, David Heckerman, and Christopher Meek. Large-sample learning of bayesian networks is np-hard. *The Journal of Machine Learning Research*, 5:1287–1330, 2004.

[16] Christos Christodoulopoulos, Sharon Goldwater, and Mark Steedman. A bayesian mixture model for part-of-speech induction using multiple features. In *Proceedings of the conference on empirical methods in natural language processing*, pages 638–647. Association for Computational Linguistics, 2011.

[17] Catherine A Cooper, Elisabeth Gasteiger, and Nicolle H Packer. Glycomod–a software tool for determining glycosylation compositions from mass spectrometric data. *Proteomics*, 1(2):340–349, 2001.

[18] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[19] Adnan Darwiche. Bayesian networks. *Communications of the ACM*, 53(12):80–90, 2010.

[20] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[21] Susan T Dumais. Latent semantic analysis. *Annual review of information science and technology*, 38(1):188–230, 2004.

[22] Raymond A Dwek. Glycobiology: toward understanding the function of sugars. *Chemical Reviews*, 96(2):683–720, 1996.

[23] Matthew Eavenson, Krys J Kochut, John A Miller, René Ranzinger, Michael Tiemeyer, Kazuhiro Aoki, and William S York. Qrator: A web-based curation tool for glycan structures. *Glycobiology*, page cwu090, 2014.

[24] Sean R Eddy and Richard Durbin. Rna sequence analysis using covariance models. *Nucleic acids research*, 22(11):2079–2088, 1994.

[25] Li Fei-Fei and Pietro Perona. A bayesian hierarchical model for learning natural scene categories. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 524–531. IEEE, 2005.

[26] Shai Fine, Yoram Singer, and Naftali Tishby. The hierarchical hidden markov model: Analysis and applications. *Machine learning*, 32(1):41–62, 1998.

[27] Nir Friedman, Iftach Nachman, and Dana Peér. Learning bayesian network structure from massive datasets: the «sparse candidate «algorithm. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 206–215. Morgan Kaufmann Publishers Inc., 1999.

[28] Zoubin Ghahramani. An introduction to hidden markov models and bayesian networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(01):9–42, 2001.

[29] David Goldberg, Marshall Bern, Bensheng Li, and Carlito B Lebrilla. Automatic determination of o-glycan structure from fragmentation spectra. *Journal of proteome research*, 5(6):1429–1434, 2006.

[30] Jorge Gracia, Vanessa Lopez, Mathieu d'Aquin, Marta Sabou, Enrico Motta, and Eduardo Mena. Solving semantic ambiguity to improve semantic web based ontology matching. 2007.

[31] TYMOTHY Grainger and Timothy Potter. Solr in action, 2014.

[32] Elias Gyftodimos and Peter A Flach. Hierarchical bayesian networks: an approach to classification and learning for structured data. In *Methods and Applications of Artificial Intelligence*, pages 291–300. Springer, 2004.

[33] ZHOU Hai-ting. Machine learning and bioinformatics. *Information and Control*, 32(4):352–7, 2003.

[34] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[35] Thomas Hamelryck. An overview of bayesian inference and graphical models. In *Bayesian Methods in Structural Bioinformatics*, pages 3–48. Springer, 2012.

[36] Sébastien Harispe, Sylvie Ranwez, Stefan Janaqi, and Jacky Montmain. Semantic measures for the comparison of units of language, concepts or entities from text and knowledge base analysis. *arXiv preprint arXiv:1310.1285*, 2013.

[37] Herlina Jayadianti, Lukito Edi Nugroho, Carlos Sousa Pinto, Paulus Insap Santosa, and Wahyu Widayat. Solving problem of ambiguity terms using ontology. 2013.

[38] Liangxiao Jiang, Dianhong Wang, and Zhihua Cai. Scaling up the accuracy of bayesian network classifiers by m-estimate. In *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence*, pages 475–484. Springer, 2007.

[39] Michael I Jordan et al. Graphical models. *Statistical Science*, 19(1):140–155, 2004.

[40] Minoru Kanehisa. The kegg database. *Silico Simulation of Biological Processes*, 247:91–103, 2002.

[41] Mohamed Korayem, Amr Badr, and Ibrahim Farag. Optimizing hidden markov models using genetic algorithms and artificial immune systems. *Computing and Information Systems*, 11(2):44, 2007.

[42] Julian Kupiec. Robust part-of-speech tagging using a hidden markov model. *Computer Speech & Language*, 6(3):225–242, 1992.

[43] Terran D Lane. Machine learning techniques for the computer security domain of anomaly detection. 2000.

[44] Pat Langley and Stephanie Sage. Induction of selective bayesian classifiers. In *Proceedings of the Tenth international conference on Uncertainty in artificial intelligence*, pages 399–406. Morgan Kaufmann Publishers Inc., 1994.

[45] Bin Ma. Challenges in computational analysis of mass spectrometry data for proteomics. *Journal of Computer Science and Technology*, 25(1):107–123, 2010.

[46] Kai Maass, René Ranzinger, Hildegard Geyer, Claus-Wilhelm von der Lieth, and Rudolf Geyer. Glyco-peakfinder–de novo composition analysis of glycoconjugates. *Proteomics*, 7(24):4435–4444, 2007.

[47] Rada Mihalcea, Courtney Corley, and Carlo Strapparava. Corpus-based and knowledge-based measures of text semantic similarity. In *AAAI*, volume 6, pages 775–780, 2006.

[48] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[49] Athanasios Papoulis and S Unnikrishna Pillai. *Probability, random variables, and stochastic processes*. Tata McGraw-Hill Education, 2002.

[50] Sangho Park and Jake K Aggarwal. A hierarchical bayesian network for event recognition of human actions and interactions. *Multimedia systems*, 10(2):164–179, 2004.

[51] Judea Pearl. Bayesian networks: A model of self-activated memory for evidential reasoning. 1985.

[52] Judea Pearl. *Markov and Bayes networks: a comparison of two graphical representations of probabilistic knowledge*. Computer Science Department, University of California, 1986.

[53] Tomaso Poggio and Federico Girosi. A theory of networks for approximation and learning. Technical report, DTIC Document, 1989.

[54] Mindy Porterfield, Peng Zhao, Haiyong Han, John Cunningham, Kazuhiro Aoki, Daniel D Von Hoff, Michael J Demeure, J Michael Pierce, Michael Tiemeyer, and Lance Wells. Discrimination between adenocarcinoma and normal pancreatic ductal fluid by proteomic and glycomic analysis. *Journal of proteome research*, 13(2):395–407, 2013.

[55] J. Ross Quinlan. Simplifying decision trees. *International journal of man-machine studies*, 27(3):221–234, 1987.

[56] Christoph Rademacher and James C Paulson. Glycan fingerprints: calculating diversity in glycan libraries. *ACS chemical biology*, 7(5):829–834, 2012.

[57] Rahul Raman, S Raguram, Ganesh Venkataraman, James C Paulson, and Ram Sasisekharan. Glycomics: an integrated systems approach to structure-function relationships of glycans. *Nature Methods*, 2(11):817–824, 2005.

[58] René Ranzinger, Stephan Herget, Claus-Wilhelm von der Lieth, and Martin Frank. Glycomedb?a unified database for carbohydrate structures. *Nucleic acids research*, 39(suppl 1):D373–D376, 2011.

[59] Irina Rish. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46, 2001.

[60] Konstantin Shvachko, Hairong Kuang, Sanjay Radia, and Robert Chansler. The hadoop distributed file system. In *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, pages 1–10. IEEE, 2010.

[61] Padhraic Smyth. Belief networks, hidden markov models, and markov random fields: A unifying view. *Pattern recognition letters*, 18(11):1261–1268, 1997.

[62] Johannes Söding. Protein homology detection by hmm–hmm comparison. *Bioinformatics*, 21(7):951–960, 2005.

[63] D Solomatine, LM See, and RJ Abrahart. Data-driven modelling: concepts, approaches and experiences. In *Practical hydroinformatics*, pages 17–30. Springer, 2008.

[64] Haixu Tang, Yehia Mechref, and Milos V Novotny. Automated interpretation of ms/ms spectra of oligosaccharides. *Bioinformatics*, 21(suppl 1):i431–i439, 2005.

[65] Christopher J Thomas, Amit P Sheth, and William S York. Modular ontology design using canonical building blocks in the biochemistry domain. *Frontiers in Artificial Intelligence and Applications*, 150:115, 2006.

[66] Ashish Thusoo, Joydeep Sen Sarma, Namit Jain, Zheng Shao, Prasad Chakka, Suresh Anthony, Hao Liu, Pete Wyckoff, and Raghotham Murthy. Hive: a warehousing solution over a map-reduce framework. *Proceedings of the VLDB Endowment*, 2(2):1626–1629, 2009.

[67] Grigorios Tsoumakas, Eleftherios Spyromitros-Xioufis, Jozef Vilcek, and Ioannis Vlahavas. Mulan: A java library for multi-label learning. *The Journal of Machine Learning Research*, 12:2411–2414, 2011.