Learning the Structure of Skills in a Professional Network Using Integer Linear Programming

John Wu
Department of Computer Science
johnwu93@gmail.com

Abstract

Bayesian networks are commonly used to represent conditional probability relationships between a set of variables in the form of a directed acyclic graph (DAG). Naturally, Bayesian Networks can be useful in modeling the skillsets of professionals on LinkedIn, since learning a set of skill can be modeled as DAG. Learning the structure of the network of skills is an important task, but it is labeled as an NP-Hard problem. In this paper, I will show how this task can be modeled as an Integer Linear Programming (ILP) Optimization Problem with exponential number of constraints. Particularly, I will show how these constraints are formed and solved. I will also compare these results with a greedy hill climbing structure learning methods and the performance of classifying the skill sets of users compared to other methods, such as Naive Bayes.

1 Introduction

Linkedin is the world's largest professional networking service that enables individuals to manage their professional network. One of the most well-know features in Linkedin is the ability for users to showcase their skills. With this feature, recruiters and future employers can directly see if an user is suited for a certain role. Also, if an user does not mentioned skillsets in their Linkedin account, it would also be useful for a future employer to determine if an user has these skillsets or has the potential to learn them effectively.

The relationship of skills can be seen as a DAG, since having a skill requires other skills and there can be no circular relationships between the requirement of learning a skill. For example, if a user have a machine learning skill, then it means the user should have strong math skills. However, a mathematician does not have to know machine learning skill. A illustration of these skills can be seen in figure 4.

Naturally, knowing the graphical structure of these skill sets is valuable since it informs users the required skillsets they need to learn a new skill and it can be easily visualized. One can manually create this model with simple intuition. However, this can be laborious and can lead to errors if there are many different skillsets. This can be even difficult for a career counselor who has working knowledge of the huge job market since they cannot take into account of the different skill sets in a market. Thus, there is a need to automatically draw a DAG of skillsets.

One approach to solve this problem is to model the structure of skills as a Bayesian Network (BN). As a Bayesian Network, one can use structured learning algorithms to generate a graphical relationship of skills and make predictions with the statistical model. In particular, I will only focus on the skillsets of Computer Scientists and Data Scientists since I am familiar with these skillsets. In the next section, I will briefly discuss how the data for this project was collected and cleaned. The bulk of section 3 will introduce Bayesian Networks and the justifications for modeling the skills problem as a Bayesian Network. Then, I will discuss how to do structured learning as an Integer

29th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

Linear Programming. Section 4 will discuss the effectiveness of these the method through several experiments. The workflow of this project can be summarized by figure 1.

2 Collecting Data/Data Processing

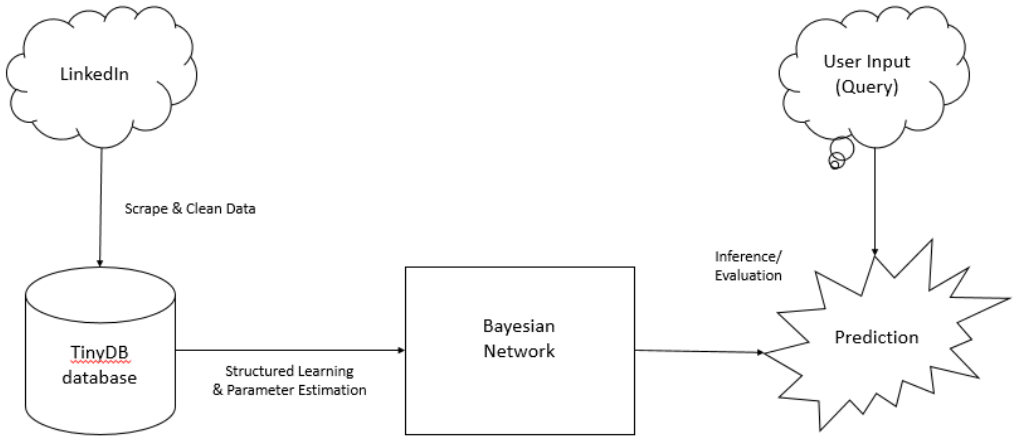

Figure 1: Project Workflow. Data is first collected online from Linkedin. Then, that information is stored into a json database called TinyDB. After preprocessing, the skeleton of a Bayesian Network is learned. Subsequently, the values parameters of the skeleton is learned. With the skill sets of a user as an input, other skills of the user can be predicted using the Bayesian Network.

Due to Linkedin's API restrictions and the unreliability of other API's, I devised my own crawler to collect data from Linkedin, which turned out to be very stable . For my crawler, I used a web testing framework called Selenium to extract the HTML layout of a user's page [1]. Selenium would extract the user's name, URL, title and list of skills into a hash map. Also, for each user, Linkedin lists other users who are similar to that user. I used this list to crawl potential users and saved their information into the hash map as user's friends. All of the users that were crawled were stored in a json database called TinyDB. The keys of the database were the user's URL.

Users' webpages were crawled in a BFS manner where the scraper uses a queue to determine which users to crawl in the future and the json database along with the queue to crawl a webpage that was not already collected. A user's information would only be crawled if the user mentioned that he/she is student, software engineer, researcher, or data scientist in their title.

In the end, I collected information of around $1,000$ users with a set of $2,000$ unique skills. Some skills, such as house gardening, would appear rarely in the dataset. Therefore, I removed the skills that were not common, only keeping the top 50 most common skills. Some skillsets were coined as different names, but could be seen as the same skill, such as `Unix` and `Linux` or `C++` and `C++ programming`. I simply renamed these instances and combined them into one to ensure that these skills refer to the same skill. It should be noted that LinkedIn has a much more sophisticated process to standardize their large skillsets dataset with millions of users [2]. The final result was a $N \times 50$ binary matrix indicating if a user has one of the fifty skills, where N is the number of users. The workflow of this part of the project can be easily summarized on the left side of figure 1.

3 Modeling Skills as a Bayesian Network

A Bayesian Network represent the joint probability distribution of a finite set of random variables through a DAG and its associated conditional probability tables. The graph constitutes the structure of the network; each node of the DAG represents a random variable. The edges of the DAG that connects two nodes implies that two nodes are conditionally dependent of each other. More

specifically, given that a set of parent nodes $Pa_1, ..., Pa_j$ directly point to a child node C with a direct edge, the conditional probability $P(C|Pa_1, ..., Pa_j)$ is directly given in a Bayesian Network and is a parameter of the model. Denoting the number of possible outcomes of a random variable Y to be $|Y|$, $P(C|Pa_1, ..., Pa_j)$ will encode $(|C| - 1)|Pa_1|...|Pa_j|$ parameters. Suppose that X represent the nodes of a Bayesian network $X_1, X_2, ...X_n$ are sorted in topological order and that $Pa(X)$ is the parent nodes of node X. Then, the joint probability of these nodes is

$$P(X_1, X_2, ...X_n) = \prod_{i=1}^{n} P(X_i|Pa(X_i)) \tag{1}$$

[3] discusses the conditional and independence properties of these Bayesian Networks in full detail.

Suppose that the joint distribution $P(X_1, X_2, ...X_n)$ were not modeled in this manner and had no conditional probability assumptions. Assuming that each state of X_i had binary outcomes, the joint probability distribution would have 2^n states and therefore would have $2^n - 1$ parameters. This means that at least $2^n - 1$ data points are required to learn the distribution, which is prohibitively expensive. However, Bayesian Networks can compact the representation of the distribution with less parameters and therefore are more favorable.

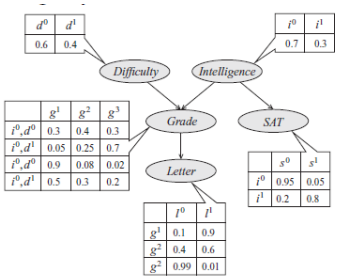

Figure 2: Example of a the classical student Bayesian Network

To illustrate these details, Figure 2 is an example of a Bayesian Network. Each node is associated to a conditional probability distribution as shown in the figure. Because of the directed edges, the joint probability distribution is represented as

$$p(d, i, g, s, l) = p(d)p(i)p(g|i, d)p(s|i)p(l|g) \tag{2}$$

If this model did not use any independence assumptions, this model would have $3 \times 2^4 - 1 = 47$ parameters, which is quite costly. (Notice that grade has 3 states rather than 2.) Using the structural information of the Bayesian network, this network would have 15 parameters and is quite expressive.

As mentioned before, the process of learning skills is naturally acyclic. A conditional probabilities table for a skill offers a nice interpretation that tells individual the probability of knowing a skill given a set of skills.

3.1 Scoring a Bayesian Network

Learning a Bayesian Network from a given dataset is essentially computing the MAP estimation of the posterior distribution, $\text{argmax}_G P(G|D)$. This is problem is essentially finding the best Bayesian DAG G that satisfies this score. This problem is equivalent to solving equation 3

$$\underset{G}{\text{argmax}}\, P(G|D) \propto \underset{G}{\text{argmax}}\, P(G)P(D|G) \tag{3}$$

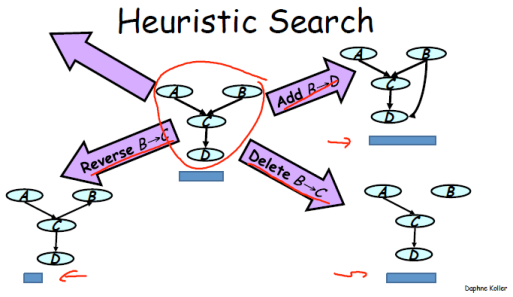

Figure 3: The operations add, delete and swap edge of Greedy Hill Climbing applied to a graph and the results

Using different assumptions can make this problem simpler. Assuming multinomiality, lack of missing values, parameter independence, ... etc., solving 3 is equivalent to maximizing the Bayesian Dirichlet (BD) score of a graph [4] which is defined as

$$BD(G) = \sum_{i=1}^{n}\sum_{j=1}^{q_i}(\log(\frac{(r_i-1)!}{(N_{ij}+r_i-1)!}) + \sum_{k=1}^{r_i}\log(N_{ijk}!)) \tag{4}$$

Adding the assumptions of likelihood-equivalence uniform Bayesian Dirichlet (BDeu) given in [4], solving equation 3 is equivalent to maximizing the BDeu scoring function, which is defined as

$$BDeu(G) = \sum_{i=1}^{n}\sum_{j=1}^{q_i}(\log(\frac{\Gamma(\frac{N'}{q_i})}{\Gamma(N_{ij}+\frac{N'}{q_i})}) + \sum_{k=1}^{r_i}\log(\frac{\Gamma(N_{ijk}+\frac{N'}{r_iq_i})}{\Gamma(\frac{N'}{q_ir_i})})) \tag{5}$$

where there are n random variables X_i, r_i is the number of bins for X_i, q_i is the number of parental instantiations of X_i, N_{ij} is the number of times where the jth parental instantiation occurs for variable X_i, N_{ijk} is the number of times $X_i = k$ given the jth parental instantiation in the dataset D and N' is the sample size. Notice that the sum of these scoring functions are also decomposible since the most-outer term is the sum of the nodes. This term represents the local score of a particular node with respect to it's parents in the graph. In this paper, we denote these local scores as $\phi(X, Pa)$, where X is its node and Pa is it's parents.

Since there are so many different configurations of a Bayesian network, it can be a difficult problem to find a Bayesian Network this score. In fact this problem is NP-hard [3].

3.2 Greedy Hill Climbing

One approach to finding structure in Bayesian Network is using Greedy Hill Climbing. Greedy Hill Climbing iteratively making a small modification to the graph G, which will generate a family of graphs $\{G'\}$, and then chooses the best G' that corresponds to the greatest increase in the scoring function. Specifically, these small modifications are adding edges to G, deleting edges from G and swapping edges from G. This can be illustrated in figure 3. We can mathematically interpret the change of add an edge $v_1 \to v_2$ as

$$\Delta(G, G') = \sum_{i=1}^{N}\phi(X_i, Pa(X_i, G \cup (v_1 \to v_2))) - \sum_{i=1}^{N}\phi(X_i, Pa(X_i, G)) \tag{6}$$

Where $Pa(X, G)$ is the set that represents the of node X in graph G. Since, only $v_1 \to v_2$ is only being affected, 6 turns out to be

$$\Delta(G, G') = \phi(v_2, Pa(X_i, G \cup (v_1 \to v_2)) - \phi(v_2, Pa(X_i, G)) \tag{7}$$

Likewise, deleting and swapping an edge is the result of eq 8 and eq 10

$$\Delta(G, G') = \sum_{i=1}^{N} \phi(X_i, Pa(X_i, G - (v_1 \to v_2))) - \sum_{i=1}^{N} \phi(X_i, Pa(X_i, G)) \tag{8}$$

$$= \phi(v_2, Pa(v_2, G) - v_1) - \phi(v_2, Pa(v_i, G)) \tag{9}$$

$$\Delta(G, G') = \sum_{i=1}^{N} \phi(X_i, Pa(X_i, G - (v_1 \to v_2) \cup (v_2 \to v_1))) - \sum_{i=1}^{N} \phi(X_i, Pa(X_i, G)) \tag{10}$$

$$= \phi(v_2, Pa(v_2, G) - v1) - \phi(v_2, Pa(v_2) + \phi(v_1, Pa(v_1, G) \cup v1)) - \phi(v_1, Pa(v_1, G))) \tag{11}$$

The problem of this approach is that it can get stuck in local minimas. To resolve this problem, one can initially start this algorithm with different initial points. Using this random restart method can be used to find new local minimas.

3.3 Integer Linear Programming

Another approach to finding structure in Bayesian Networks is using Integer Linear Programming. An ILP problem is a an optimization problem where the objective is to maximize a linear objective function with affine constraints and that the variables of the problem take integer constraints. This problem is posed as equation

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & \mathbf{c}^T \mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ \text{and} \quad & \mathbf{x} \in \mathbb{Z}^n \end{aligned} \tag{12}$$

To solve problem, it first usually requires relax the problem into a Linear Program (LP) without the integrality constraint. Thus the problem becomes

$$\begin{aligned} \underset{x}{\text{maximize}} \quad & \mathbf{c}^T \mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \end{aligned} \tag{13}$$

This problem can be easily solved using a numerical optimization method, such as the Simplex method, and there are many software programs that uses these techniques to solve these problems efficient, which includes CVX [5].

Using the solution from LP relaxed problem, the ILP problem can then be solved using different methods, such as branch-and-bound. For branch-and-bound, branch-and-bound is a recursive algorithm that solves its current optimization problem by solving two sub problem that includes the problem that it is trying to solve. In the context of ILP, if a variable x_i is not an integer solution to the problem, but is close to a integer value l, the problem will be broken down into two problems: one problem posed as the current problem with the added with the constraint $x_i \leq l - 1$ and the other problem with the added with the constraint $x_i \geq l$. Of the two sub-problems, branch-and-bound will select the sub-solution with the largest maximum value.

3.4 Structured Learning as an ILP

In order to learn the structure of a Bayesian Network as an ILP problem, it is important to decide the variables that need to be solved for ILP and their possible values. For a Bayesian Network, we

can represent the edges of the network through the binary variable, $I(W \to v)$, which is 1 if a set of nodes W are all parents for the v and 0 otherwise [6]. For Figure 2, the binary values that will have the value as 1 is $I(\{\emptyset\} \to d)$, $I(\{\emptyset\} \to i)$, $I(\{d,i\} \to g)$, $I(\{d,i\} \to g)$,$I(\{i\} \to s)$ and $I(\{l\} \to g)$, while all other combination of variables will be 0. The BDeu score equation (5) can be rewritten as an affine combination of binary variables $I(W \to v)$. Thus, the objective function that is being maximized is

$$\sum_{v,W} c(v,W) I(W \to v) \tag{14}$$

Where $c(v,W)$ is coefficient of the BDeu score of $I(W \to v)$.

One of the obvious constraints to this problem is that each node v must belong to only one set of parent W. [6] This problem can be written as:

$$\forall v \in V : \sum_{W} I(W \to v) = 1 \tag{15}$$

Also, we can see that for an acyclic graph, for all subsets C in a graph, there exists a $v \in C$ that has no parents in C. In other words, for all subsets C in a graph, there exists a $v \in C$ that only has a parent in the complement of C, W. This can be be written as the constraint

$$\forall C \subseteq V : \sum_{v \in C} \sum_{W \cap C = \emptyset} I(W \to v) \geq 1 \tag{16}$$

With the objective function (14), the linear constraints (15, 16) and the restriction that $I(W \to v)$ must either 0 or 1, the main problem can be formed as ILP problem. Obviously, this problem has exponential number of constraints and variables. [7] discusses heuristics to solve this problem by iteratively finding nodes that would be sink nodes to the graph.

3.5 Parameter Estimation in Bayesian Networks

After finding a structure that best represents the data, the next step is to compute the parameters of the Network. Specifically, the parameters are the CPD tables of the Network. A common problem in Statistics is to compute the best joint distribution $p_\theta(x)$, parameterized by θ, where data points are drawn from a distribution p^*. An approach to solving this problem is solving the maximum likelihood problem which is defined as

$$\max_\theta \frac{1}{|D|} \sum_{x \in D} \log p_\theta(x) \tag{17}$$

By eq 1, eq 17 can be written as

$$\begin{aligned} \max_\theta \frac{1}{|D|} \sum_{x \in D} \log p_\theta(x) &= \max_\theta \frac{1}{|D|} \sum_{n=1}^{N} \sum_{i=1}^{|V|} \log p(x_i^n | x_{pa(i)}^n ; \theta) \\ &= \max_\theta \frac{1}{|D|} \sum_{i=1}^{|V|} \sum_{i=1}^{N} \log p(x_i^n | x_{pa(i)}^n ; \theta) \end{aligned} \tag{18}$$

Solving this problem analytically, it can be easily shown that the parameters of the CPD is in the form [8]:

$$\hat{\theta}_{ijk} = \frac{N_{ijk}}{N_{ij}} \tag{19}$$

where $\hat{\theta}_{ijk}$ is the one of the CPD values for node i where it takes the assignment of jth parent configuration i and has the value k.

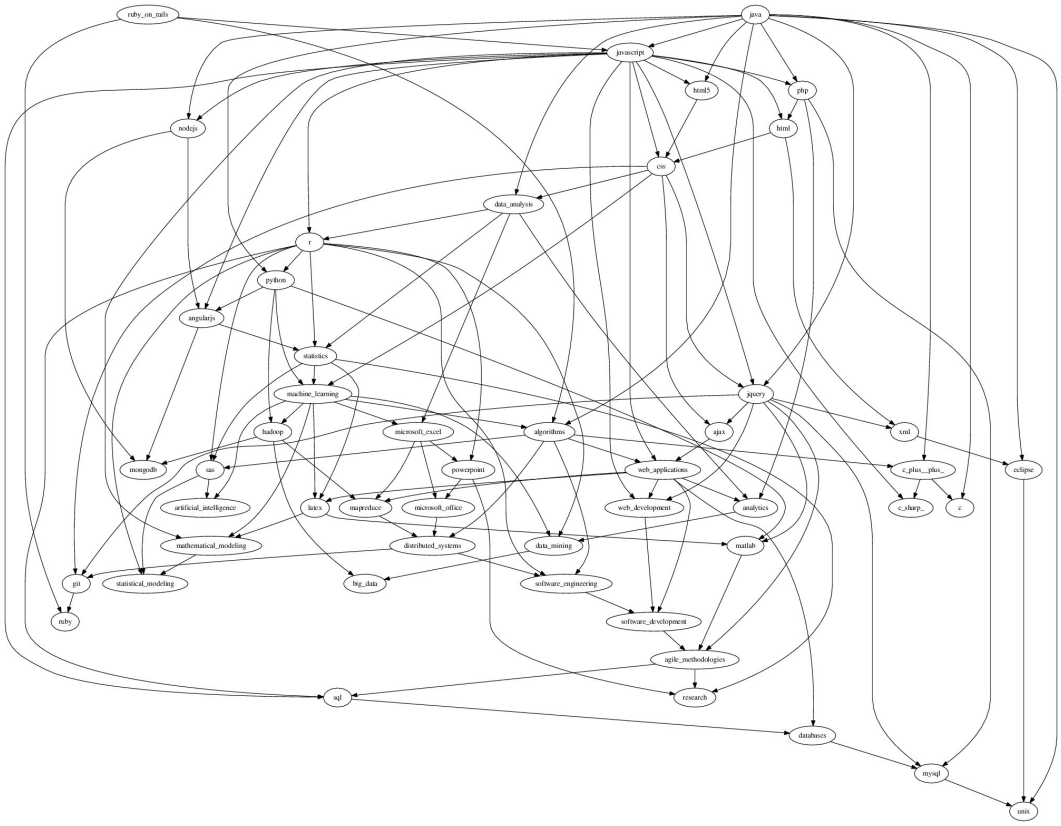

Figure 4: Structured Learning Using the ILP Method

4 Experiment and Results

To experiment the effectiveness of these two models, I split the LinkedIn dataset into a training and test set, where the training set contains about 800 users and the test set contains 200 users.

In order to learn the Bayesian Network of the Linkedin dataset, I used a software package called Gobnilp [9] that uses the ILP method discussed in the previous section to learn the structure of dataset. The objective function is the BDeu score (eq. 5). In order to learn the Bayesian Network of the Linkedin dataset with the ILP method, I constrained the the number of parents for each node to have at most 3 to just cut the computation time. I used the software package Gobnilp to setup and solve the ILP problem and include the number of parents constraint [9]. Also, I used Python program called Pebl to learn the structure of the network using a greedy hill climbing [10]. This package scores the network by using BD metric (eq. 4). Due to the time constraints, I did not modify the package to make the scores be the same. For this experiment, the number of random restarts that was performed was 100.

The graph corresponding to the structured learning method of ILP is in figure 4. The graph corresponding to the greedy method is in figure 5. Empirically, it looks like the ILP method shows an intuitive placement of the skillsets. In the top, we can see that Java is one of the roots of the network, which is a fundamental skill that many professionals first pick up when pursuing a career in Computer Science or Data Science. Also, it is interesting to see that many skills that were associated to a particular formed a cluster. We can see the nodes on the left side of the graph are skills involving statistics and mathematics and these skills directly connect other. Likewise, the right side of the graph shows relationships of software engineering skills. It is interesting to see that there was an edge connect CSS to Machine Learning. When computing the CPD values of Machine Learning, we see that if CSS is 0, then there is a high probability that user would know Machine Learning. Likewise if CSS is 1, then there is a high probability that user would not know Machine Learning.

Figure 5: Structured Learning Using the Greedy Method

Model	BD	BDeu
ILP	**-5078.14**	**-4527.02**
Greedy	-5119.80	-4695.09
b1	-5888.91	-5326.72
b2	-5830.51	-5240.99
b5	-6683.37	-8124.32
b8	-18012.41	-56293.08

Table 1: Performance of the structure scores of different models

For the greedy method, we can see that the edges is less intuitive. It is interesting to see that the root of the tree is `web development`, which is typically not one of the first tools that a computer scientist and data scientist learns. It is also interesting to see that the edges of the graph give similar clusters as the one shown for the ILP method.

To numerically compute the quality of these two structures, we use the metrics $BD(G, D)$ (eq. 4) and $BDeu(G, D)$ (eq. 5). On table , we can see that ILP method outperforms the greedy method for both methods. This may be due to the fact, that for the formulation of the greedy method is not a concave problem and it got stuck in many local maximum. Where as the relaxation of the ILP lead to a local maximum which helped achieve a better score. The scores can be summarized in 4.

Additionally, I compared the models with the simple graphs that are variants to the graph of Naive Bayes models. For Naive Bayes, the graph is represented as a class C that points to all of the other nodes. An illustration of a Naive Bayes graph is figure 6

This graph is commonly used to compute the MAP of $P(C|X_1, X_2, ...)$ and it turns out that this simple graph is very good in prediction. The model was modified so that rather than having one node pointing to multiple nodes, it would be a set of N nodes that would be the parent node of all the other nodes. The number of nodes were arbitrarily selected as 1, 2, 5 and 8. We denote these graphs as the baseline graphs or $b1$, $b2$, $b5$ and $b8$.

The baseline graphs were use to to further verify if it is necessary to use structured learning to learn the structure of skills. It turns out these Naive Bayes models performed worse compared to the other

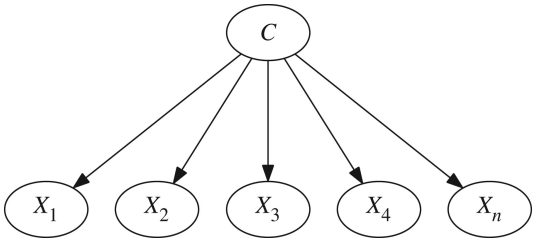

Figure 6: Figure of a Naive Bayes Model. If the graph had multiple nodes being predicted, then instead of having one node, C, point to all the other nodes, we would have multiple nodes pointing to the other nodes.

Model	Acc_1	Acc_2	Acc_5	Acc_8
ILP	0.66	**0.61**	**0.45**	0.28
Greedy	0.66	0.59	0.42	**0.29**
b1	**0.67**			
b2		0.55		
b5			0.36	
b8				0.26

Table 2: Accuracy of Structured Learning Models

models. In fact, the BD and BDeu scores decreases as the number of nodes increases. This would make sense because we are creating awkward graphs that do not represent the structure of skills.

Along with the quality of the structure of the data, it is also important to see the predictive performance of these models. For this experiment, I only computed the MAP of a query given that all the other skillsets were given. Specifically, the queries that I wanted to predict were the following:

1. [`Sql`]
2. [`Sql Node.js`]
3. [`Sql Node.js R Ruby Statistics`]
4. [`Sql Node.js R Ruby Statistics Hadoop Ajax Java`]

The baseline models were configured so that the parent nodes are the nodes listed above and the direct children of these nodes are not listed. Baseline model b_i only made predictions with queries of length i. This is mainly due to the fact that this Naive Bayes are designed to make effective predictions. Suppose that the ground truth for user i is defined as y_i and MAP prediction given that his skillsets e is defined as $\hat{y}_j = \text{argmax}_x \, P(x|e)$ For this experiment, I used Acc_j and $TotalAcc_j$ to denote the quality of the prediction where j is the number of desired queries I want to predict. They are defined as the following:

$$Acc_j = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i == y_i) \tag{20}$$

$$TotalAcc_j = \frac{1}{Nj} (\sum_{i=1}^{N} \sum_{k=1}^{j} (\hat{y}_{ik} == y_{ik}) \tag{21}$$

From table 4, we can see that when having a query of length 1, the baseline model slightly outperforms the other models. This due to the fact that Naive Bayes models are designed for predictions while structured learning algorithms are not. Then we can see that as the length of the queries increases, the Acc_i decreases. This may be due to the fact that the baseline does not manipulate the independence assumptions of the data correctly. It can also be shown that ILP is the best predictive model for queries of length 2 and 5. However, Greedy slightly beats ILP with a large number of queries. I am not sure why this is happening since the structure scores of ILP outperforms Greedy.

Model	$TotalAcc_1$	$TotalAcc_2$	$TotalAcc_5$	$TotalAcc_8$
ILP	0.6623	**0.7857**	0.8389	0.8377
Greedy	0.6623	0.7835	**0.8398**	**0.8398**
b1	**0.6667**			
b2		0.7511		
b5			0.8009	
b8				0.8290

Table 3: Total Accuracy of Structured Learning Models

Additionally, I obtained the similar results for the $TotalAcc_i$. Instead of total accuracy decreasing as the length of the query increases, it increases. This is due to the fact that $TotalAcc_i$ is not a harsh metric compared to Acc_i and that these models have a hard time computing $\hat{y}$ perfectly. It is interesting to see that the Greedy very slightly outperforms the other methods when the query is equal or greater than 5.

5 Conclusion

In this paper, I showed that we can automatically obtain an effective and intuitive Bayesian Network to represent the skillsets of Computer Scientists and Data Scientists by forming the structured learning problem as an ILP problem. It turns that this model's graph outperforms other graphical models in terms of structure and prediction.

In the future, it would be interesting to explore why the Greedy method outperforms ILP for prediction tasks when the length of the query is large. it would be nice to see how the greedy and ILP methods compare if both were trained using BDeu as scoring metric. Also, it would be interesting to explore the use of having hidden nodes in the graph, where the hidden node is not a skill in the dataset. For the context of the LinkedIn dataset, I suspect that the graph can inductively identify if a user is a software engineer or a data scientist. The significance of this is that it can remove many unnecessary edges. This problem was explored in [11] and solves this problem using the EM algorithm.

The use case of finding a graphical structure of skillsets is significant for LinkedIn as it aligns to LinkedIn's business goal of finding connections of skills and to the entire world.

References

[1] Unmesh Gundecha. *Selenium Testing Tools Cookbook*. Packt Publishing Ltd, 2012.

[2] Mathieu Bastian, Matthew Hayes, William Vaughan, Sam Shah, Peter Skomoroch, Hyungjin Kim, Sal Uryasev, and Christopher Lloyd. Linkedin skills: large-scale topic extraction and inference. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 1–8. ACM, 2014.

[3] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[4] Alexandra M Carvalho. Scoring functions for learning bayesian networks. *Inesc-id Tec. Rep*, 2009.

[5] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, March 2014.

[6] Mark Bartlett and James Cussens. Advances in bayesian network learning using integer programming. *arXiv preprint arXiv:1309.6825*, 2013.

[7] Mark Bartlett and James Cussens. Integer linear programming for the bayesian network structure learning problem. *Artificial Intelligence*, 2015.

[8] Sontag David. Lecture 3: Learning in probablistic graphical models. 2014.

[9] James Cussens and Mark Bartlett. Gobnilp 1.6. 2 user/developer manual1. 2015.

[10] Abhik Shah and Peter Woolf. Python environment for bayesian learning: Inferring the structure of bayesian networks from knowledge and data. *Journal of machine learning research: JMLR*, 10:159, 2009.

[11] Gal Elidan, Noam Lotner, Nir Friedman, Daphne Koller, et al. Discovering hidden variables: A structure-based approach. In *NIPS*, volume 13, pages 479–485, 2000.