DISCOVERING HIDDEN COGNITIVE SKILL DEPENDENCIES BETWEEN KNOWLEDGE UNITS USING MARKOV COGNITIVE KNOWLEDGE STATE NETWORK (MCKSN)

A dissertation submitted
to Kent State University in partial
fulfillment of the requirements for the
degree of Doctor of Philosophy

by

Fatema Nafa

November 2018

Dissertation written by

Fatema Nafa

B.Sc., Attahadi University, Libya, 2002

M.S., Attahadi University, Libya, 2006

Ph.D., Kent State University, USA, 2018

Approved by

Dr. Javed I. Khan ________________________, Chair, Doctoral Dissertation Committee

Dr. Austin Melton ________________________, Members, Doctoral Dissertation Committee

Dr. Feodor Dragan ________________________

Dr. Arvind Bansal ________________________

Dr. Kambiz Ghazinour ________________________

Dr. Katherine Rawson ________________________

Dr. Phillip Hamrick ________________________

Accepted by

________________________, Chair, Department of Computer Science

________________________, Dean, College of Arts and Sciences

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

DEDICATION

This dissertation is dedicated to my always-encouraging, faithful parents, my brilliant, loving, and supportive husband, Dr. Salem Othman, and our sweet and beautiful kids, Ahmed, Alla, and Aya. Thank you for your love, support, and encouragement.

ACKNOWLEDGEMENTS

Thank you, GOD, for giving me the power and faith. Thank you for giving me that knowledge to complete this dissertation. I'd like to express deepest appreciation and gratitude to my advisor, Dr. Javed Khan, who is the reason this dissertation exists. Thank you for your good advice, for everything you've taught me, and most of all for being a steady ally throughout my Ph.D.

I'd also like to thank the professors who have served in my doctoral committee, such as Dr. Feodor Dragan, Dr. Austin Melton, Dr. Arvind Bansal, Dr. Kambiz Ghazinour, Dr. Katherine Rawson ,and Dr. Phillip Hamrick. I would like to individually thank every single person who participated in the studies presented here and thank the algorithm lab at Kent State University's Computer Sciences department, especially Dr. Arne Leitert for his help.

My deepest gratitude goes to my second family at the Department of Computer Science, particularly for Marcy Curtiss for having an open heart and Janet Katila for her kindness and wisdom. Their dedication to their job is admirable, and they always do it without waiting for recognition or reward. They have taught me a lot.

As graduate students we're not supposed to have much of a social life, but luckily, I've found friends who helped me directly or indirectly during my Ph.D. I wish to thank my best friend, Dr. Amal Babour, for being a wonderful study partner. She has always been up for an adventure, and for sharing every sense with me. My sincere thanks to my Friend Maha Allouzi for her help and support.

I owe the greatest thanks to my patient husband, Dr. Salem Othman. His unconditional love and support has kept me on task and is my true motivation. My God blesses my life by making me a mom for three adorable kids, Ahmed, Alla, and Aya. They were very patient when I was very busy. Thank GOD that I have them in my life.

I am also thankful to my mother and my father. Without their incredible love, guidance, and emotional and financial support, I would not be where I am. Additionally, I would like to thank all my brothers and sisters for their support and encouragement.

Fatma Nafa

November 2018, Kent, OH

LIST OF ABBREVIATIONS

NOTATIONS	MEANING
C_i	A concept *i*
SKM	Semantic Knowledge Map
BL_k	Bloom’s Taxonomy Levels *k*=1,2,3,4
ϕ_i	A logical Relationship between a set of Cognitive Skill Dependencies
BT	Bloom Taxonomy
RBT	Revised Bloom Taxonomy
CSBT	Computer Science-based Bloom Taxonomy
CSD	Cognitive Skill Dependency
NLP	Natural Language Processing
SVD	Singular Value Decomposition
WN	WordNet
VN	VerbNet
KU	Knowledge Unit
$B(V_i)$	Bloom Taxonomy Verb
$V(d)$	Variable Distance between Two Verbs
£	Fleiss' kappa
FOL	First Order Logic
S_i	Skill Inference Rule (*SIR*) Strength
MN	Markov Network
MB	Markov Network Blanket
MLN	Markov Logic Network
MCKSN	Markov Cognitive Knowledge State Network
MAP	Maximum a Posteriori Query
CP	Conditional Probability
MCMC	Markov Chain Monte Carlo
IRB	Institutional Review Board
β_m	*MCKSN* Model Threshold
β_h	Human Model Threshold
$H_b[i, s]$	Human Model Matrix where *i*: is the Relationship Index, *s*: is the Human Model Index, and *b*:is the Cognitive Skill Dependencies Index.
M_b	*MCKSN* Model Estimation Result
H_b	Human Model Estimation Result
H_b^{mm}	Scaled Human Model Data using MinMax scaler
M_b^{mm}	Scaled *MCKSN* Model Data using MinMax scaler
H_b^{LG}	Scaled Human Model using Log scaler

M_b^{LG}	Scaled *MCKSN* Model using Log scaler
H'^{mm}_b	Human Model Data using MinMax scaler
M'^{mm}_b	*MCKSN* Model Data using MinMax scaler
$H_b'^{LG}$	Human Model using Log scaler
M'^{LG}_b	*MCKSN* Model using Log scaler
$A_{i,j}$	Performance Matrix using MinMax Scaled Data with $a_{i,j}$ Elements
$B_{i,j}$	Performance Matrix using Log Scaled Data with $b_{i,j}$ Elements.
TP	True Positives
TN	True Negative
FP	False Positives
FN	False Negative
P	Precision using the MinMax Scaling Technique
P'	Precision using the Log Scaling Technique
R	Recall using the MinMax Scaling Technique
R'	Recall using the Log Scaling Technique
D	Accuracy using the MinMax Scaling Technique
D'	Accuracy using the Log Scaling Technique
F-measure	Accuracy Measure using the MinMax Scaling Technique
F'*-measure*	Accuracy Measure using the Log Scaling Technique
ME	Margin of Errors
CL	The Confidence Level
SE	The Standard Error

CHAPTER 1: INTRODUCTION

1.1 Introduction

This chapter reviews the key features of this dissertation and provides the reader with the necessary background knowledge to grasp the potential impact and significance of the dissertation as a whole. The contents are broadly sketched out to give a good overview of what will be accomplished later on in this dissertation, with a particular focus on motivation, dissertation objectives, problem description, proposed solution, assumption, application of the proposed model and dissertation structure.

1.2 Motivation

Cognitive psychology has observed that there are two primary mechanisms of the mental process: knowledge structure, and the process of using knowledge. The organization of knowledge plays an essential role in both understandings the text and to facilitate learning. Merrill in 1987 recommended "The purpose of instruction is to promote that active cognitive processing that best enables the learner to use the most appropriate cognitive structure in a way consistent with the desired learned performance." (Reigeluth, 2013)

Organization of knowledge matters in learning any topic. Each learning regimen is written in a certain way to present each knowledge unit in specific order. This is done to maximize the understanding of the knowledge contents. In fact, some textbooks unable to accomplish this with high accuracy are often judged as incompetent. Although they may contain all the needed concepts which interact the reader to pick the textbook to read, they may not be well-written, making the concepts more difficult to comprehend. So, the organization and the presentation of the knowledge units in the textbook are more necessary than the concepts themselves. For example, consider a group of first graders getting their first mathematics lesson. If the instructor chooses to start with “fractions” without teaching the subtraction, addition, and the multiplication, the students will be unable to understand the lesson itself. So, instructional organization is important in such a manner to sequentially build up the knowledge base for a student for further instruction. The text’s form is illustrated by the quality of content organization. A very well-organized text will obviously help in understanding the concepts. One of the most apparent problems that a typical faculty member must focus on includes which domain concepts to teach, and how to rank each domain concept or teaching method for the level of thinking regarding cognitive skills. An early, widely used set of categories was proposed by (Bloom, Engelhart, Furst, Hill, & Krathwohl, 1956).

Knowledge structure refers to the interrelationships among knowledge components. In this dissertation, the interconnections among knowledge units are represented as a well-known cognitive theory called Bloom’s Taxonomy (*BT*). Proposed by Benjamin Bloom, it is a modern concept that is used as a guideline for educators to develop teaching regimens, organize learning goals, and create assessments (T. Thompson, 2008) and (Lister, 2000). Bloom’s Taxonomy places learning objectives into three domains: cognitive, psychomotor, and effective (Bloom, 1956). The cognitive domain is related to the knowledge and mental skills of a learner. It is the most widely used domain, including six levels from low to high mental (processing) levels.

Bloom's Taxonomy was modified by Anderson (Anderson et al., 2001), who made a significant alteration to it by adding and ordering the names of each level. However, the number of levels was kept consistent. The revised cognitive domain levels (from simplest to most complex) are: 1) Remembering; 2) Understanding; 3) Applying; 4) Analyzing; 5) Evaluating; and 6) Creating (as in Figure 1.1). The educationist (1993) argues the importance of the distinctions in the cognitive levels rather than the hierarchy of the levels; "the categories themselves are not independent but interdependent"(Crossland, 2010). As a result, it is imperative to use the hierarchy for the learning regimen and learning objectives.

The revised Bloom Taxonomy was modified for the domain of Computer Sciences (Nafa & Khan, 2015). However, the number of levels had changed. The levels of Computer Science Bloom Taxonomy (*CSBT*), from simplest to most complex, are: 1) Understanding (BL_1); 2) Applying (BL_2); 3) Evaluating and Analyzing (BL_3); and 4) Creating (BL_4) (as in Figure 1.1). *CSBT* provides a more flexible structure, facilitating the classification of the knowledge domain. The main goal for creating a revised version is to provide an effective order of *BT* cognitive skills for computer sciences. *CSBT* introduces a useful specific-hierarchy to the existing Bloom's Taxonomy.

The revised Bloom's Taxonomy is a cognitive skills Taxonomy that has been applied for different educational purposes in many fields of study. In the field of Computer Science, Bloom's Taxonomy has been used in course design, teaching methodology, material preparation, and measuring student responses to learning (Doran & Langan, 1995) and (Oliver & Dobele, 2007). The *ACM* Computer Science Curriculum specifies learning objectives based on the revised version of Bloom's Taxonomy (Parham, Chinn, & Stevenson, 2009). There is a strong need to describe Computer Science knowledge units regarding learning goals and regarding levels of mastery.

The following will therefore define concrete objectives and a clear framework in order to bring the techniques one step further towards the application of Bloom's theory.

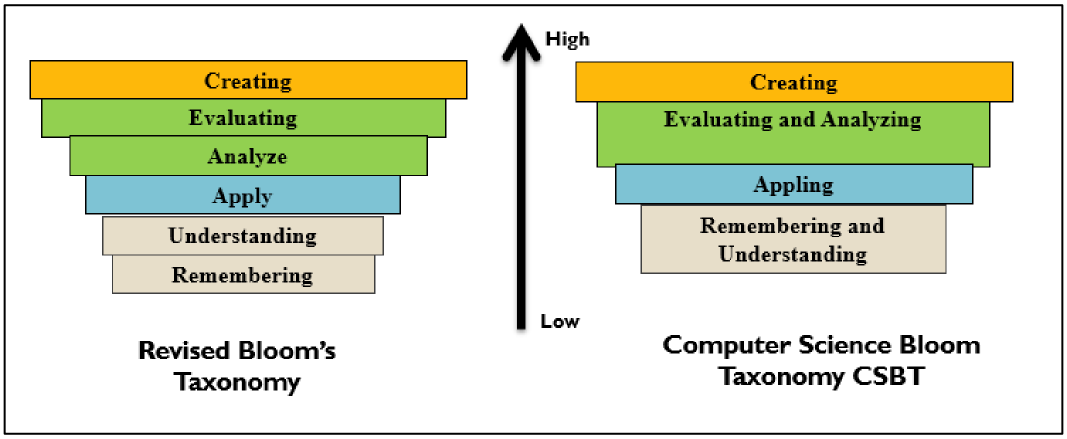


Figure 1.1. The Changes from the Revised Bloom's Taxonomy to the Version of Computer Science Bloom Taxonomy (CSBT).

1.3 Dissertation Objectives

The primary objective behind this work is mainly driven by the expanding interest and a recent increase in research based on the provided resources of using the Bloom's Taxonomy theory in the field of computer science. This dissertation provides a framework for transferring the learning process from quantity to quality regarding *CSBT* cognitive theory. Also included is the realization of whether a model concerning the presentation of the learning materials of computer science can be built to assess the learner with regards to cognitive *CSBT* theory, as well as how those regimens connect in a specific domain space.

In order to address this issue, several areas of investigation such as WordNet (*WN*), VerbNet (*VN*), Singular Value Decomposition (*SVD*), First Order Logic (*FOL*), Skill Inference Rules (*SIR*), Markov Logic Network (*MLN*), Markov Cognitive Knowledge State Network (*MCKSN*), and Probability-Based Inference, were included in this research.

1.4 Key Terminology

This section defines some of the main terms that are used throughout this dissertation.

Concept (*C*): The Concept is the smallest unit in knowledge representation. It captures a knowledge domain that is eventually acquired by learning the concepts in it and their complex interrelationships. Normally, a knowledge domain has a terminology whose specific semantics are understood by the domain experts.

Knowledge Unit (*KU*): A Knowledge Unit is a highly interrelated set of concepts which are dense and semantically dependent. Often the understanding of a concept in a *KU* is mutually enhanced by the other concepts in that *KU*. It can consist of two or more concepts.

Cognitive Skill (*CK*): Cognitive skills are human skills of information processing. This includes knowledge gained and understood because of thinking, experience and sensations. Cognitive abilities involve knowledge banks, attention, memory (including working memory), making and evaluating solutions, reasoning, estimating, problem solving, decision making, understanding, speaking and speech understanding skills (Anderson et al., 2001).

Cognitive Skill Taxonomy: It is a classification of cognitive skills and strategies that develop from a complex set of life-long learning skills in cognition. The classification system can be categorized into different types known as Bloom's Taxonomy(*BT*) (Bloom's, 1965), Revised Bloom's Taxonomy (*RBT*) (Anderson et al., 2001), and Computer Science-based Bloom's Taxonomy (*CSBT*), the latter being a modification of the Bloom's Taxonomy system which is more useful to computer science learners than existing generic ones (Nafa & Khan, 2015).

WordNet Relationships (*WN*): It is a dependency relationship between concepts, it has different types which are (hyponym, hypernym, meronym, and holonym).

A Semantic Knowledge Map *SKM= (C, E)* is defined as a graph where, $C = \{c_i\}$ is the set of learning concepts c_i, and a set of edges $E = \{e_{ij}(ci,cj,BL_k)\}$ is the Cognitive Skill Dependencies (*CSD*) between the concepts ci and c_j at a *CSD* level k (BL_k) per the cognitive skill taxonomy. The taxonomy identifies relationship models using different skill levels. The classical Bloom's Taxonomy has six levels, while the Anderson's revised Bloom's Taxonomy has six types but with modified semantics. The six types of cognitive skills are as follows: {BL_1= Remembering, BL_2= Understanding, BL_3= Applying, BL_4= Analyzing BL_5= Evaluating, BL_6= Creating}, whereas the Computer Science-based Bloom's Taxonomy (*CSBT*) has four skill levels such as {BL_1= Understanding, BL_2=Applying, BL_3= Analyzing-Evaluating, BL_4= Creating}.

Skill Inference Rules (*SIR*): An *SIR* is defined as a logical relationship between a set of Cognitive Skill Dependencies (*CSDs*) $d_i=e\ (a_i,\ b_i,\ BL_i)$. The logical relationship between any set of *CSDs* can be expressed as a First Order Logic expression. More formally, an example of this is $\phi_i = \forall_{A,B,\ C} \{e(A,\ B,\ BL_i) \wedge e(B,\ C,\ BL_i) => e(C,\ A,\ BL_i)\}$. In other words, if concept *A* is needed to learn concept *B*, and concept *C* is needed to learn *B*, then concept *C* is needed to learn concept *A*.

Markov Cognitive Knowledge State Network (*MCKSN*): An *MCKSN* is defined as an undirected graph $G= (F,\ R)$, where each node F_i in the node set *F* represents a Cognitive Skill Dependency at a given Bloom level. $R \subseteq F \times F$ is a set of edges connecting the nodes. Each edge $r_{i,j}$ represents the appearance of a Cognitive Skill Dependency in the same Skill Inference Rule (*SIR*).

1.5 Problem Description

Given a Semantic Knowledge Map $SKM = (C,\ E)$, a subset of known Cognitive Skill Dependencies (*CSD*) is found between the concepts with their level specifications

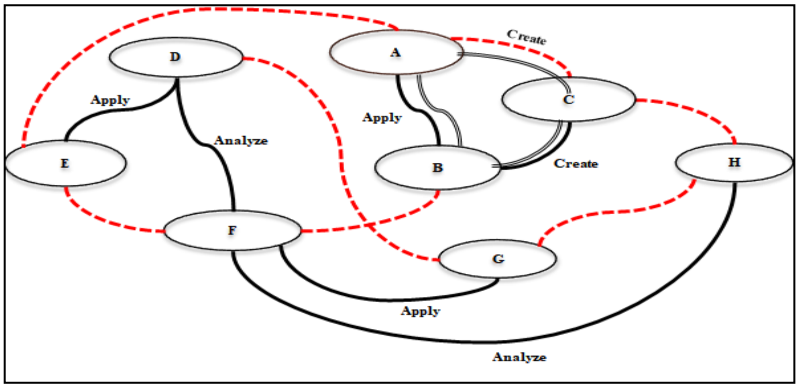


Figure 1.2. Example of BL$_{k,}$ SIR, and CSD in Semantic Knowledge

BL$_k$, and a set of Skill Inference Rules ϕ_i. Find out the remaining *CSDs*. The graphical representation denotes the symbols; the *BL$_k$* (in bold), the *SIR* (in double), and the *CSD* (in the dot), are shown in Figure 1.2.

Consider a real example, in this scenario suppose that a learner needs to learn some concepts from Algorithm book related to different topics such as *{Graph, Graph-Traverse, BFS, Binary-tree, Data-structure, Algorithm, Insertion-sort, and Heap-sort}*. Some of the Cognitive Skill Dependencies (*CSDs*) between concepts are given which are {*Apply, Analyze, and Create*}in addition, a skill dependency among the given *CSDs* encompassed in the *SKM*. Consider that the learner starts with three concept *{Graph, Graph-Traverse, BFS}* which are three nodes in the *SKM* as in Figure 1.3. As the *CSDs* between concepts shows that a concept '*Graph'* is needed to be known to Apply concept '*Graph-Traverse*' and concept '*Graph-Traverse*' is needed to Create a concept '*BFS*'. The question is can we recommended that the learner should learn a concept '*Graph*' to Create a concept '*BFS*'.

This scenario represents a subgraph of the *SKM* as in Figure 1.3. Consist of some CS-concepts with their *CSDs*.

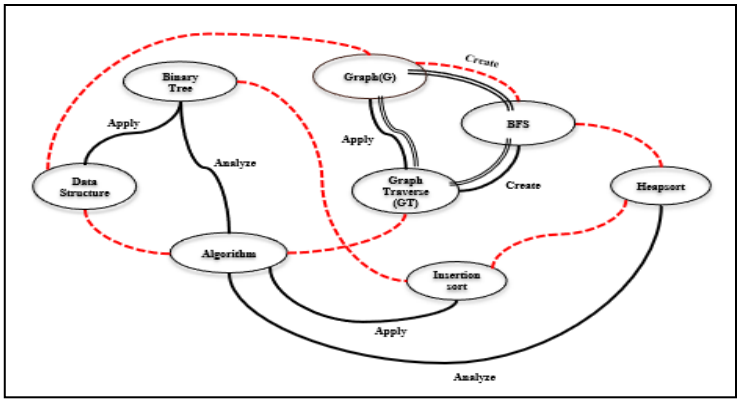


Figure 1.3. An Example of Some Cognitive Skill Dependencies between CS-Concepts.

1.6 Proposed Solution

To answer the question above, a human may easily estimate it either by using common sense and their experience or by asking domain experts. However, a computer requires an immense amount of knowledge to reason out the relationships of different areas. To access the best answer, a computer requires algorithmic techniques that can process the information for building the model. However, it is not trivial to build a model that can answer the question like a human. Therefore, a major challenge is to build a model that can assist the learner. To answer the question, Markov Cognitive Knowledge State Network (*MCKSN*) model was used.

Definition*:* A Markov Cognitive Knowledge State Network (*MCKSN*) is defined as an undirected graph *G= (C, E),* where each node ci in the node set *C* represents a

Cognitive Skill Dependency (*CSD*) at a given Bloom level. $E \subseteq C \times C$ is a set of edges connecting the nodes. Each edge $e_{i,j}$ represents the appearance of *CSD's* in the same Skill Inference Rules (*SIR*) (as illustrated in Figure 1.4).

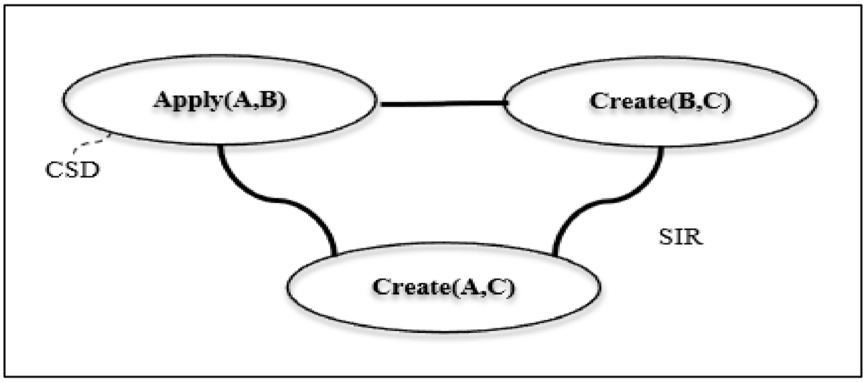


Figure 1.5 **M**arkov **C**ognitive **K**nowledge **S**tate **N**etwork (**MCKSN**).

1.7 Problem Formulation for Applying *MCKSN*.

Given a Markov Cognitive Knowledge State Network (*MCKSN*), subset of *CSDs*, and set of Skill Inference Rules (*SIR*) and assumed to be true. Estimate the probability of the inferred *CSDs* to be true.

The proposed model contains some key ingredients which are relied upon in each part of the dissertation. The main contribution is to introduce a mental blueprint, using these key ingredients to create a proper solution and to proof how these key ingredients can be coupled together to tackle different angles of the research problem. The three keys ingredients of the presented model are:

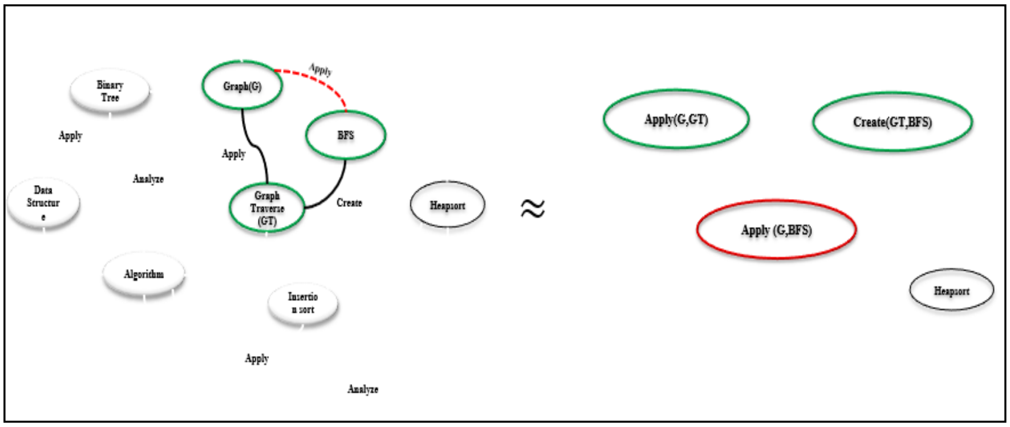


Figure 1.4. Sub Graph of SKM Converted to MCKSN.

- Mapping the *SKM* into *MCKSN*,
- Using human knowledge to describe the Skill Inference Rules (*SIR*) among the *CSDs* via First Order Logic (*FOL*), and
- Using the Probability Graphical Inference to infer *CSDs*.

The act of mapping *SKM* into *MCKSN* through *CSDs* and *SIR* among the cognitive relationships, where the facts describe CSDs between two nodes in the *SKM* and the Skill Inference Rules (*SIR*) describe *CSDs* among the *CSDs*, forms a clique in the *MCKSN*.

The Skill Inference Rules (*SIR*) among *CSDs* is expressed as First-Order Logic (*FOL*) rules. An example of this is to suppose a subgraph from the previous case includes three nodes *('Graph', 'Graph-Traversal', and 'BFS').* Based on Bloom's taxonomy of learning theory, a learner is expected to learn those concepts at different Bloom levels. In other words, [If a concept 'Graph' is needed to be known to '*Apply'* concept 'Graph-Traverse,' and concept ‘*Graph-Traverse*’ is needed to be known to ‘*Create*’ concept ‘*BFS*’ then a concept ‘*Graph*’ is needed to be known to ‘*Create*’ a concept ‘*BFS*’].

There is no magic coding recipe for taking a sentence expressed in natural language or any other form and showing it in first-order logic. However, the syntactic restrictions of first-order logic must be obeyed. By obeying First Order Logic’s syntax, the above sentence can be expressed as a Skill Inference Rules (*SIR*) as follows:

$$\mathit{Apply}\ (G, GT) \wedge \mathit{Create}\ (GT, BFS) \Rightarrow \mathit{Apply}\ (G, BFS)$$

The dissertation shows how the inference of *CSDs* can be recast in a probabilistic setting. The aim is to apply a probabilistic graphical model to infer *CSDs* between concepts jointly.

These three ingredients are the general pillars of the proposed model. Each of them, however, has a number of sub-domains which are the tunable components making the model flexible. In Chapter Three, the first component is explained in detail, followed by a proper discussion. In Chapter Four, the second component is described in detail. In Chapter Five, the last component is explained and followed with an example. The experimental studies aim to showcase how the model can contribute to the research problem. Therefore, the proposed model grants new perspectives and brings about balance by showing the inference of *CSDs* among concepts in a CS domain.

1.8 Subtasks to Support the Main Contribution

To support the dissertation, an argumentation is articulate around one central question (linked to Chapter Five), and the sub-research questions (strictly related to Chapter Three and Four of this dissertation), as introduced below.

Sub Questions: The sub-questions arise as challenging, due to the central question. Also, dealing with text involves inherent semantic ambiguities in natural language, as well as touching the cognitive skills. Questions regarding these issues include the following:

- Are there any cognitive verbs that can describe a specific domain?
- Is Bloom’s measurable verbs list indicative of the cognitive skills, and how can the other verbs which are not on Bloom’s verbs list be identified based on cognitive skill levels?
- From a practical point of view, how is it possible to logically design a language to map *SKM* and its internal relationships with Cognitive Skill Dependencies?
- How can the difficulties of learning a new topic be simplified by designing a learning map through the revised Cognitive Theory?
- How can an automatic tool be designed for the management of future ACM/IEEE CS curricular revisions (which are expected to have a continued emphasis on Bloom’s Taxonomy)?

The necessity of inferring the Cognitive Skill Dependencies will help learners to connect the knowledge gaps between learning objectives and the learning regimen based on each learner's specific cognitive level. This also effectively and efficiently accomplishes the aim of improving critical thinking abilities for learners. The lecturer would also have a congruent understanding of learning objectives and learning regimen, teaching students how to master all new concepts in each knowledge unit. Finally, the learner's mental skill levels should increase as they progress from one topic to the next.

A meta learning recommended model was proposed as an application of this work. The application developed to explore an interesting angle by looking for the intersection between logic and probability in one dimension. The next section will describe the general structure of this proposed application of this work which is meta learning recommended model.

1.9 Application of *MCKSN* Model

In order to achieve the study's objectives and to help to answer its research questions, a novel meta learning recommended model was developed as an application that relies on several components which will be introduced progressively. Figure1.6 illustrates an overview of the proposed application. The model involves several basic linguistic preprocessing tasks (as illustrated in Figure 1.6).

The First phase involves basic linguistic preprocessing techniques such as Text-Preprocessing, Natural Language Processing (*NLP*), Domain Specific Extraction, and Semantic Relationship Extraction. In Text-Preprocessing, the model assumes that the input files are in the plain text format. All other formats are turned into plain text before beginning the other steps. Next, Natural Language Processing incorporates *NLP* tools such as splitting each sentence into a sequence of tokens (where tokens are unique concepts). The Stanford Parser, which is used to parse each sentence to get its part-of-speech (verb, noun, adjective, etc.) is used to extract semantic relationships between concepts (Nafa, Khan, Othman, & Babour, 2016c).

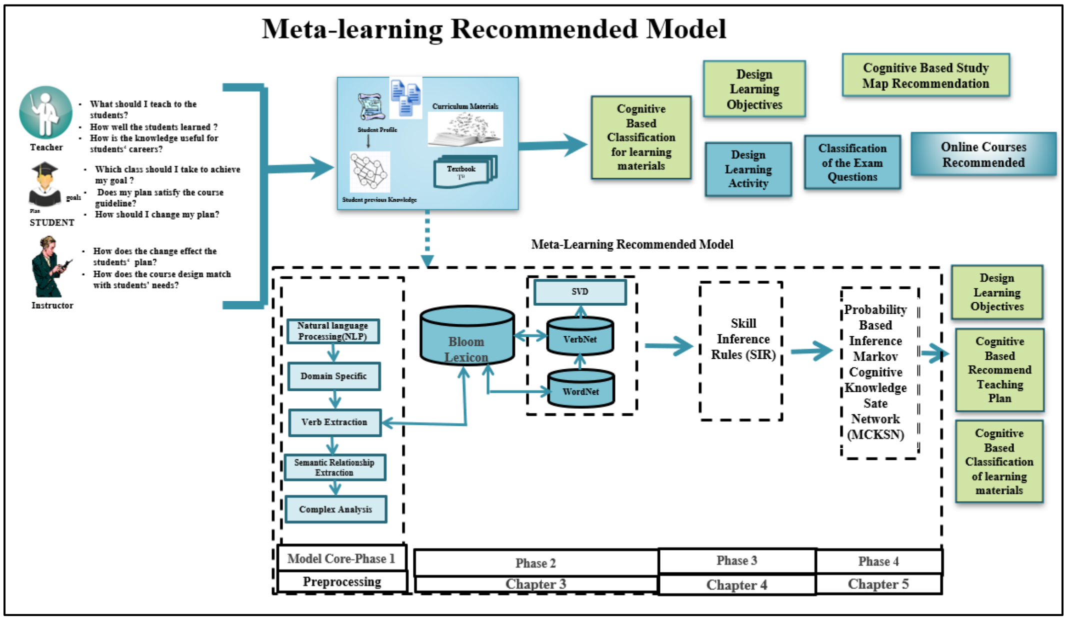
This phase also presents the Computer Science Bloom Taxonomy (*CSBT*), which is implemented using a scheme-based analysis of a revised version of Bloom' Taxonomy. This in turn provides specific cognitive levels for Computer Science learners inspired by (Nafa & Khan, 2015). The model core component plays a very important role in this design. It is the first step for preprocessing the textbook and Bloom's Taxonomy, the final form of which is represented as a semantic Knowledge Map (*SKM*) and a revised version of cognitive levels (*CSBT*).

The Second Phase of this model is classifying the verbs based on their Cognitive. Bloom's Taxonomy provides a ready-made structure and list of action verbs. These verbs are the key to extract the Skill Inference Rules (*SIR*). However, Bloom's original list of verbs was limited; not all verbs are included in the list. All the verbs in this list are action verbs since the learning objectives are concerned with what the students can do at the end of mastering a specific knowledge unit. This component proposes three techniques: WordNet, VerbNet, and Singular Value Decomposition (*SVD*) (Nafa, Khan, & Othman, 2017) . The details of this phase can be found in Chapter Three.

The Third Phase answers the central question of this dissertation. This phase investigates the application of Markov Cognitive Knowledge State Network (*MCKSN*) technique with human evaluation of the inferred Cognitive Skill Dependencies (*CSD*). The model is a probability-based inference. The details of this phase are presented in Chapter Five followed by the results and the human evaluation in chapter Six.

The idea of this dissertation is to build a model that ascertains the Cognitive Skill Dependencies between existing concepts in a specific domain. It can then assess the learner in the mastery of any knowledge unit at each cognitive learning level. The idea addresses it from an innovative angle, guided along the way by the study's research questions.

This work's contribution can be articulated around the central themes, giving a complete model introduced through the chapters of this dissertation.



Meta-learning Recommended Model
Teacher
What should I teach to the students?
How well the students learned ?
How is the knowledge useful for students' careers?
STUDENT
goals
Plan
Which class should I take to achieve my goal ?
Does my plan satisfy the course guideline?
How should I change my plan?
Instructor
How does the change effect the students' plan?
How does the course design match with students' needs?
Student Profile
Curriculum Materials
Student previous Knowledge
Textbook
Cognitive Based Classification for learning materials
Design Learning Objectives
Design Learning Activity
Classification of the Exam Questions
Cognitive Based Study Map Recommendation
Online Courses Recommended
Meta-Learning Recommended Model
Natural language Processing(NLP)
Domain Specific
Verb Extraction
Semantic Relationship Extraction
Complex Analysis
Bloom Lexicon
SVD
VerbNet
WordNet
Skill Inference Rules (SIR)
Probability Based Inference Markov Cognitive Knowledge Sate Network (MCKSN)
Design Learning Objectives
Cognitive Based Recommend Teaching Plan
Cognitive Based Classification of learning materials
Model Core-Phase 1
Preprocessing
Phase 2
Chapter 3
Phase 3
Chapter 4
Phase 4
Chapter 5

Figure 1.5. The Overall Architecture of the Proposed Model.

1.10 Assumptions

The proposed techniques used to solve the problem contain a few assumptions that need to be addressed. They are as follows:

- Verbs typically indicate semantic relations between concepts.
- Concept is the smallest unit in knowledge-unit. It is an element of a knowledge domain that is eventually acquired by learning the member concepts and their complex interrelationships.
- In simple sentence, verbs typically indicate direct semantic relations between subject and object.
- In complex sentence, concepts that nearest to the verb are semantically related.
- We only consider knowledge-units of Computer Science domain.
- The text used for extracting the relationships is structured, and in English

1.11 Dissertation Structure

This dissertation presents a meta learning recommended model foundation at the algorithm and technical level to support the above goals. Chapter Two presents the interpretations of the related work. The model is divided into phases (as illustrated in Figure 1.6). Tasks are offered throughout the chapters in detail. The second phase of the model, classification methodologies for the cognitive verbs, is presented in Chapter Three, the third task of the model shown in Chapter Four introduces the construction of Skill Inference Rules (*SIR*). Chapter Five puts into practice the lessons learned from the previous Four chapters. It gives particular attention to the central question of this dissertation after answering each sub-question. It introduces a technique to infer Cognitive Skill Dependencies (*CSD*), and it puts the tasks together in order to conclude exciting results armed by the human judge for the final results.

The dissertation ends with a conclusion of the main findings and outcomes, including tentative answers to the study’s research questions, along with a discussion of this dissertation’s limitations. Finally, it discusses how this model could profitably be generalized and transposed to other languages and application domains.

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

The work presented in this chapter is situated at the intersection of several areas of related prior work: The Natural Language Processing (*NLP*), Linguistics, and Knowledge Object and Mental Models. Each of these are discussed in turn. Figure 2.1 illustrates a general view of the literature review work.

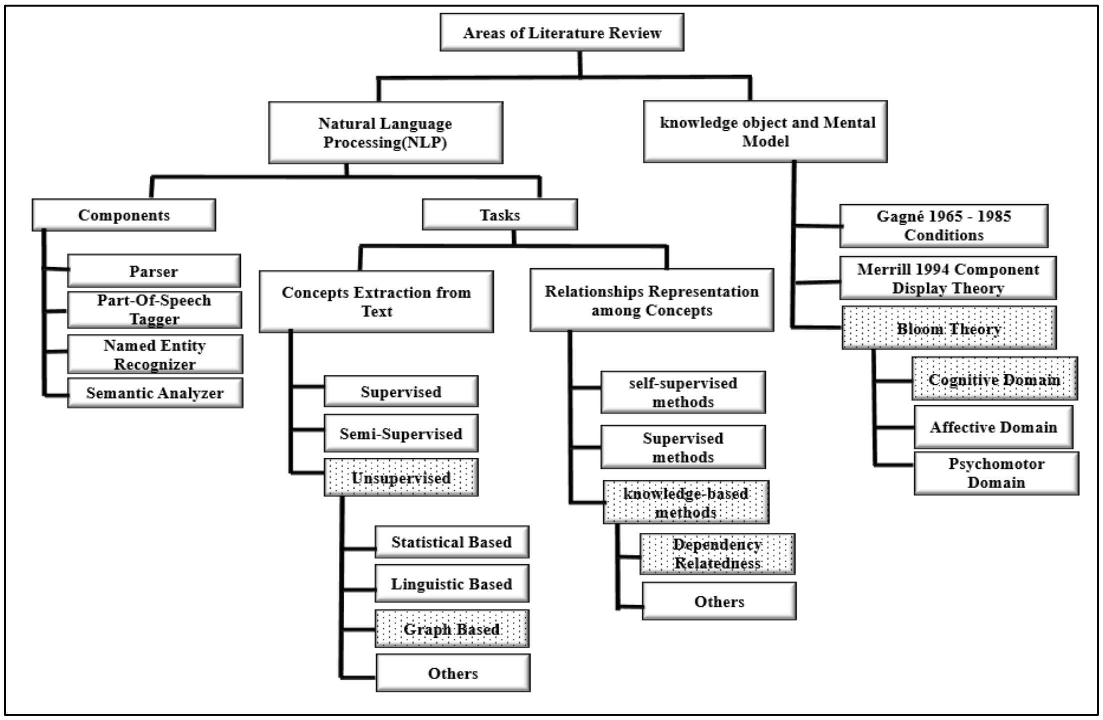


Figure 2.1. Three Different Area of the Literature Review

2.2 Natural Language Processing (*NLP*)

Natural Language Processing (*NLP*) is a branch of linguistics, artificial intelligence, and Computer Science. Its purpose is to develop a computer program that can generate text in natural language and speech patterns. The goal is to enable computers to communicate with humans in the same way that humans communicate with other humans (C. D. Manning, Manning, & Schütze, 1999). *NLP* has different research areas, some of which interrelate with the component and other items in the tasks of *NLP* in the area of the concept extraction, which is a part of Information Extraction. *NLP* has various techniques used for concept extraction including supervised, semi-supervised, and unsupervised techniques (Tur, Hakkani-Tür, & Schapire, 2005). One effective technique for concept extraction is a graph, very effectively enabling exploration of the concepts. More about graph representations of text is discussed in (Valle & Ozturk, 2011) (Walter, 2004) (Mishra, Huan, Bleik, & Song, 2012). In a graph, representation concepts are represented by nodes, and relations between concepts are represented by edges. A graph-based approach is used for the extraction of concepts from the textbook because it is very accurate and efficient for domain-specific tasks. This study focuses on textbooks since they provide a comprehensive list of domain concepts, and extracted concepts are domain specific. Therefore, using a graph-based approach is a promising method to meet the study’s demands.

In the area of the relation extraction, different categories of methods as presented in Figure 2.1. This work uses knowledge-based techniques where the relation is a dependency relation among concepts, and the graph is a representation of the extracted relationship. Research exists concerning graphical text representations such as concept graphs and ontology (Mitra, Wiederhold, & Kersten, 2000). Concept Graph Learning is also proposed to present relations among concepts from prerequisite relations among courses. Using graph-based method is a promising way to answer the study's questions and to discover novel cognitive relationships between knowledge units.

2.3 Linguistics

Verbs are central to the syntactic structure and semantics of a sentence. Existing computational resources and classifications developed for verbs can be classified into these three types:

Syntactic Resources: Examples of these are complex dictionaries (Adorni & Zock, 1996) and (Kipper, Korhonen, Ryant, & Palmer, 2006) and are mostly manually developed. An entry here has verb forms and subcategorization information.

Semantic Resources: Examples of these include FrameNet (Baker, Fillmore, & Lowe, 1998) and (Shi & Mihalcea, 2005) and WordNet(G. Miller, 1998). FrameNet groups words according to conceptual structures and their patterns of combinations. On the other hand, WordNet groups words into synsets (synonym sets) and records semantic relations between synsets. However, little syntactic information is present in these resources. According to (Kipper, Dang, & Palmer, 2000), WordNet lacks generalization, and its level of sense distinction is too fine-grained for a computational lexicon.

Syntactic Semantic Resources: Here, verbs are grouped by properties such as shared meaning components and morpho-syntactic behavior of words in Levin's 1993 verb classification. Since then, VerbNet (Kipper et al., 2000) has expanded this classification with additional verbs and classes.

Even though an extensive collection of literature exists on verb classification, none of the presented techniques have been developed to classify the verbs based on Bloom's Taxonomy levels. Benjamin Bloom and his colleagues provided the verbs to help identity which action verbs align with each Bloom level to describe the learning objectives (Starr, Manaris, & Stalvey, 2008). Benjamin Bloom provides a sub-list; not all verbs are included. There is a need in the computer sciences to use the domain verbs in order to keep the description of the learning objectives measurable and clear.

2.4 Knowledge Object and Mental Models

According to this study's research, no previous works have investigated three different taxonomies to represent the three domains of learning: a cognitive Taxonomy focused on intellectual learning, an effective Taxonomy concerned with the learning of values and attitudes, and a psychomotor Taxonomy that addresses the motor skills related to learning. One of the cognitive taxonomies is known as Bloom's Taxonomy (Bloom's, 1965). Bloom's Taxonomy has been applied in the field of computer science. Specifically, such taxonomies have been used in four different ways: 1) course design, 2) teaching methodology, 3) the creation of learning and evaluative materials, and 4) student responses to learning activity (Buck & Stucki, 2001), (Scott, 2003), and (E. Thompson, Luxton-Reilly, Whalley, Hu, & Robbins, 2008).

This section appraises the work of many research projects that have applied Bloom’s Taxonomy to the field of computer science. Machanick presents the idea of ordering materials according to the required cognitive skills taught within three computer science courses (Machanick, 2000). Bloom’s Taxonomy was used to assign grades in an introductory programming course. Grading was based on a Bloom-level mastery of tiered curricular components rather than grading on a curve by (Starr et al., 2008) .In a review of their work, the Taxonomy for computer science was questioned (Johnson & Fuller, 2006). The issue with Machanick’s method is that exams regularly fail to test the knowledge of students for each level of mastery in Bloom’s Taxonomy (Scott, 2003). Because of this, teachers cannot accurately assess the depth of mastery for individual students. A proposed solution was to use Bloom’s Taxonomy to assess the cognitive difficulty of computing courses in an IT program by formulating and calculating a Bloom Rating (Crowe, Dirks, & Wenderoth, 2008). A Bloom level was assigned to each test question according to the level of cognitive behavior required to properly answer it. Based on the above work, by using a Bloom Rating, a Bloom-based course assessment tool could be constructed and deployed in a second-level programming course (Schulte & Bennedsen, 2006). The result is the assignment of a grade that is based on objective measurements of learning outcomes. The paper describes the cognitive tasks required at each of the three grade tiers. Finally, Manaris et al. (Lister & Leaney, 2003) applied BT within CS to specify learning objectives of human-computer interaction courses. They presented a collection of courses for various target audiences, including freshman non-majors, junior/senior majors, and graduate students. For each course, they provided an outline containing learning objectives using BT, the amount of time to be spent on each topic and related in-class activities. Closely related research was also done by Thompson et al.; their focus was on Computer Science assessment (E. Thompson et al., 2008). Their main goal was to use Bloom's Taxonomy to assist in designing introductory programming examinations. More recent research done by Starr et al. focused on specifying assessable learning objectives in Computer Science (Starr et al., 2008). They believed that their idea of integrating Bloom's Taxonomy with Computer Science curriculum created more effective faculty communication and strengthened the department's assessment program. Other research work completed for specific Computer Science areas of education that used Bloom's Taxonomy includes a test-driven automatic grading approach for programming (Hernán-Losada, Pareja-Flores, & Velázquez-Iturbide, 2008) , Bloom's Taxonomy levels for three software engineer profiles (Bourque, Buglione, Abran, & April, 2003) , and Bloom's Taxonomy for system analysis workshops (Khairuddin & Hashim, 2008).

2.5 Summary and Discussion

The use of existing taxonomies is not as efficient for computer science. This study addresses a novel aspect of the problem. Based on David Kolb's research (Kolb, 2014), it is apparent that different people can enter the learning cycle at different points. A revised BT is modified to show how *CSBT* cognitive thinking would be more applicable to computer science than the existing generic ones. In “Conceptual Knowledge Space,” written by Javed I. Khan, Yongbin Ma, and Manas Hardas (Khan & Hardas, 2007), the authors demonstrated how courses can be composed, based on knowledge ontology. It presented a novel methodology to evaluate the bottom-up technique for teaching programming concepts based on the theory of constructivism from educational psychology. Educators in teaching employed their technique; students do not employ or are not able to employ the bottom-up to the technique of constructing concepts in learning. Most of the previous work does not focus on building automatic models to assist in analyzing domain concepts, where the domain concepts are identified with a cognitive link between all the concepts. The link is a verb identifying the certain skill necessary to be learned in the prerequisite concept and the target concept to be achieved at a certain skill.

A review of related work is presented in this Chapter. Next Chapter gives special attention to classify Computer Sciences verbs into their cognitive levels using three different methodologies.

CHAPTER 3: EXTENDING COGNITIVE SKILL CLASSIFICATION OF COMMON VERBS IN THE DOMAIN OF COMPUTER SCIENCES

3.1 Introduction

This chapter presents an important part of the picture: a classification of the verbs into cognitive skill levels. For this task, not all verbs are equally important; this study particularly focuses on identifying verbs indicative of idenifying cognative skill level linked to the domain of computer sciences.The classification of a domain-specific verb is defined as a Cognitive Skill connection between concepts used in sentences with a given verb. The chapter introduces three different techniques used to classify verbs, whereas Bloom's verb list is used as a baseline method to classify some of the CS verbs. but not all verbs are included in Bloom's verb list. This poses an interesting question: What about the other verbs which are not on Bloom verbs list?

The techniques include WordNet (G. Miller, 1998), which was used to access the verb synonym; VerbNet, used to access the verb class; and Singular Value Decomposition (*SVD*), for all other verbs not included in WordNet and VerbNet. The three techniques will be explained in greater detail in this chapter.

The techniques are based on linguistic dictionaries for verbal classification. It is ready to use.This work made easy access by coding them and investigating the contents of those dictionaries. This work adds a contribution by serving the cognitive area via using WordNet and VerbNet lexical databases.

A running example of the verb classification is shown in Figure 3.1. The textbook is organized into a hierarchical structure, where the organization is represented as chapters, sections, sub-sections, paragraphs and sentences. The smallest level in the hierarchy is the sentence. The structure of the sentences differs throughout the book, depending upon the need for each. This study classifies sentences into four different structural types: simple sentences, compound sentences, complex sentences, and compound-complex sentences. The study's approach assumes that concepts which are semantically related tend to be "near" in a plain text. This assumption arises from the principle of coherence on linguistics (Foltz, Kintsch, & Landauer, 1998). Based on this assumption, the study's proposed technique is applied to knowledge units extracted from the texts in order to discover semantic relations between concepts. Moreover, this approach is able to find the Cognitive Skill levels between concepts based on the verb.

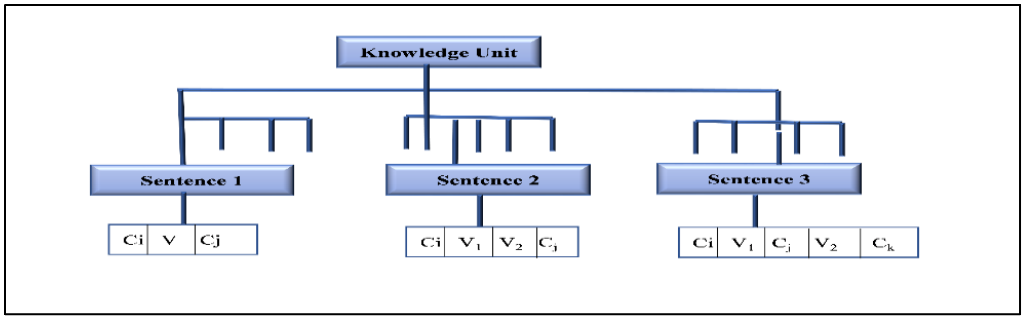


Figure 3.1. Sentence Structure

The Cognative Skill levels between concepts in the sentences are captured from different sentences structures. One such example is a specified knowledge unit about the '*Heapsort Algorithm'* to extract the concepts and the verbs from given knowledge unit. This study's major goals include the following: Finding the low-level concepts and the high-level concepts and showing the dependencies between the concepts based on the verb is accomplished in the first phase of the model, where the output is represented as a Semantic Knowledge Map (*SKM*). Next, high-level concepts are used to describe the learning objectives based on the verb cognitive level, where the verb level is unknown. However, this study's proposed techniques are able to figure out these verb levels as sub-tasks.

The given knowledge unit includes some concepts: *Heap-Sort, heap-property, time, priority-Queue, max-heap, producer, sorting, array*}. The process of finding the Cognitive Skill levels of the verb is done by describing the learning objectives required for mastering this knowledge unit at different cognitive levels.

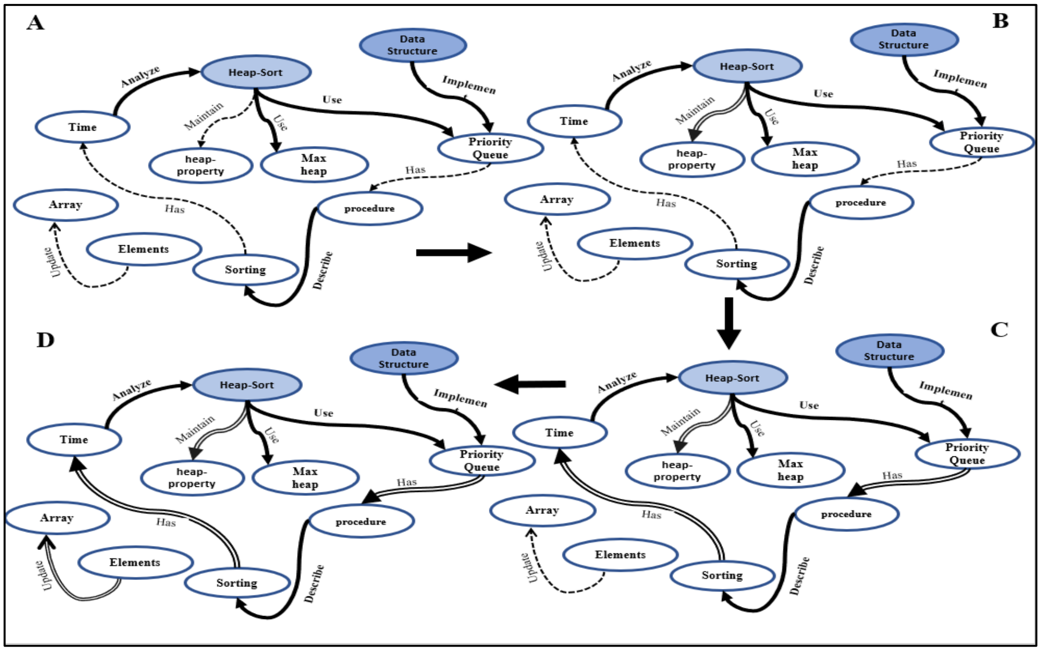


Figure 3.2. An Example of a Known and Unknown Bloom's Verbs.

In Figure 3.2 (A), only five verbs are known in their cognitive levels from Bloom's original list; they appear as dark black lines in the graph. In Figure 3.2 (B), by using the first methodology for verb classification (*WordNet*), only one verb classified into its cognitive level; the verb appears with a double line in the graph. In Figure 3.2 (C), by using the second methodology for verb classification (*VerbNet*), only two verbs classified into its cognitive level; the verbs appear with a double line in the graph. In Figure 3.2 (D), by using the third methodology for verb classification (*SVD*), the rest of the verbs are classified into their respective cognitive levels; the verbs appear with a double line in the graph. After all classification methods are Applied, verbs are classified into their cognitive levels, and the concepts are edited to descibe the learning objectives for this knowledge unit. The teacher can then ask him/herself what cognitive levels are necessary for his/her students to master this knowledge unit.

3.2 Bloom's Verbs and their Cognitive Skill Levels

Bloom's Taxonomy provides a ready-made structure and list of action verbs. These verbs are the key to writing learning objectives. However, Bloom's original list of verbs was limited; not all verbs are included in the list. All the verbs are action verbs since the learning objectives are concerned with what the students can do at the end of mastering a specific knowledge unit. As an example, a list of the active verbs used to assess a Remembering level is shown in Figure 3.3.

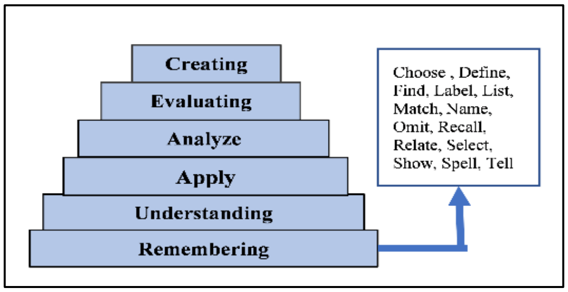

Creating
Evaluating
Analyze
Apply
Understanding
Remembering
Choose , Define, Find, Label, List, Match, Name, Omit, Recall, Relate, Select, Show, Spell, Tell

Figure 3.3. Example of Bloom Action Verb List Remembering Level.

To run the linguistic analysis for the knowledge unit in the textbook, Stanford University's Core *NLP* library is used (C. Manning et al., 2014). This step has been done in the first phase of the study's model, but more characteristics are added to analyze the verbs in the knowledge units. The results of the sentence structures for the verbs have also been analyzed. The most common modification used to get a high accuracy for the results involves incorrect POS tags; errors are shown as stemming; and sometimes a verb can be mistagged as a noun. These incorrect POS tags, causing incorrect parsing structures, are modified manually. All auxiliary verbs are also removed by checking the verb with a list of all auxiliary verbs and their derivatives. For more accurate results, the proposed techniques are introduced and used (*WordNet*, *VerbNet*, and *SVD*). The three techniques will be explained in detail respectively.

3.3 Extension Technique based on WordNet (*WN*)

An assumption used in this study is the following: **If** there is a *WordNet* relation between two verbs, **THEN** the Bloom label is there;

$$\textbf{If}\quad V_i \leftrightarrow V_j \;\; \textbf{\textit{Then}} \;\; \beta(V_i) = \beta(V_j)$$

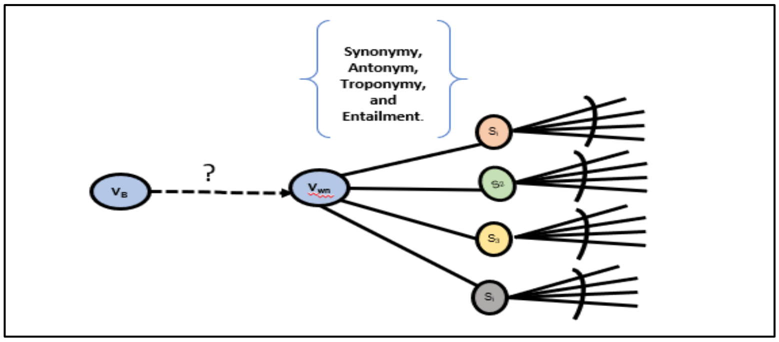


Figure 3.4. The Verb Relationship in WordNet

WordNet documents the verbs based on 14 different files (Klavans & Kan, 1998), each file covering a semantic domain: verbs of bodily care and functions, change, cognition, communication, competition, consumption, contact, creation, emotion, motion, perception, possession, social interaction, and weather verbs. Each of these files has a “unique” set of beginners which correspond to the top most verbs in that hierarchy. These sets also denote the most basic concept in that tree, which is specialized by the remaining verbs in that tree as an example of the WordNet verb relationship shown in Figure 3.4.

The verbs provide most of the semantic frame of sentences and are considered the most important syntactic category. Although each syntactically correct sentence must have a verb, they do not necessarily require a noun. This study is especially interested in the verbal relation of synonyms. These synonym relations have more expressive power and are better tailored for the task of the Cognitive Skill Levels. Synonymy relation is at the base of WordNet’s structure, along with being the most important Levels found in WordNet (which is already implicit in the notion of a synset). WordNet-like taxonomies behave in some ways like a dictionary, and in others as an ontology. To avoid Performance, WordNet in this research is used as a dictionary for verb synonym relations. Around 3,600 verb senses are included in WordNet.

```
Def WordNet Technique ( ):
Input BT-Verb [ ], CS-Verb [ ]
Output: BTN-Verb [ ], Unknown-Verb [ ]
Def Find-Bloom-Level (CS-Verb-Synonym):
    For Verb in CS-Verb-Synonym:
        For Verb1 in BT-Verb:
            If Verb in BT-Verb ():
                BT-Verb [Verb]=Level
                BTN-Verb [Verb]. Append (Level)
            Else:
                Unknown-Verb [Verb]. append (Zero)
                BTN = GetmostFrequ (BT-Verb)
Return BTN-Verb [ ], Unknown-Verb [ ]
Def Find-WordNet-Synonym (CS-Verb):
    For pos in poses:
        For Synset in WN. Synsets (CS-Verb, pos):
            For lemma in synset. Lemmas ():
                If Name! = CS-Verb and Name not in syns:
                    Syns.append(Name)
Return CS-Verb-Synonym
```

Figure 3.5. WordNet Technique Algorithm.

As the first technique for finding the level of domain-specific verbs based on Cognitive Skill Dependencies, all domain-specific verbs are mapped to their verb synonyms from the WordNet database. However, WordNet has a few limitations, one of which is its limitation of not having all the classes for all verbs, classifying some of the verbs but not the others. It also does not cover special domain words, nor does it include forms of irregular verbs.

Figure 3.5 presents an algorithm used in WordNet technique. Input for the algorithm includes the original Bloom verb list *$(B(V_i))$* and the domain-specific verb list (V_j). An algorithm starts by reading a domain-specific verb list and checks which verbs are in Bloom's verb list; it returns two lists-known verbs as known in Bloom's list and as unknown as Bloom's list. It then starts to maintain the unknown verb list by checking verb synonyms from the WordNet database; some new verbs have been added to known verbs as unknown in Bloom's list. In case the verb synonym does not return any Bloom level for the verb, the Algorithm returns a new verb list with Bloom's classification and another verb list not in Bloom's Taxonomy. Thus, new verbs synonyms have been added to the known verbs as those in Bloom's list. The list will be saved as unknown verbs in Bloom's Taxonomy. A limitation of WordNet includes gaps between verbs in the database; for that reason, some of the verbs will not be found in the WordNet database (K. J. Miller, 1998). Finally, for those verbs in which a classification is not found, the algorithm starts the classification process over for verbs but uses a different methodology using the VerbNet methodology. This will be explained in detail in the following section.

3.4 Extension Technique based on VerbNet (*VN*)

VerbNet (*VN*) is a vast online repository for the classification of English verbs (Schuler, 2005). It includes syntactic and semantic information for classes of English verbs derived from Levin’s Classification (as explained in Related Works, Chapter Two). It is an updated version that is considered more detailed than the version included in the original organization. *VN* classification considers very important properties such as the lexical meaning of a verb and the kind of argument interchanges that can be observed in the sentences with the verb. The classification of VerbNet is based on the senses of verbs. It covers 5,200 verb senses. The classification is partially hierarchical, including 237 top-level classes with only three other levels of subdivision (Schuler, 2005).

The VerbNet database also contains information about the correspondence between the classes of verbs and lexical entries in other resources. Each verb class in *VN* includes a set of members, thematic roles for the predicate-argument structure of these members, sectional restrictions on the arguments, and frames consisting of a syntactic description and semantic predicates with a temporal function. New subclasses are added to Levin’s original classes to achieve syntactic and semantic coherence among members.

VerbNet is a rich database with verb classification, providing easy access for use by the programming language. It has been used to help *NLP* applications such as semantic role labeling (Swier & Stevenson, 2004) and word sense disambiguation (Dang, 2004). However, it is not very helpful when it comes to processing texts in specific domains where verb senses only partly overlap with those in general language use.

```
Def VerbNet Technique ( ):
Input NBT-Verb [ ], CS-Verb [ ]
Output: $BTN_2$-Verb [ ], Unknown-Verb [ ]
Def Find-Bloom-Level (Verb-VN-Category ):
    For Verb in Verb-VN-Category:
      For Verb in NBT-Verb:
        If Verb in BT-Verb ():
          BT-Verb [Verb]=Level
           $BTN_2$-Verb [Verb]. Append (Level)
        Else:
           Unknown-Verb [Verb]. Append (Zero)
            $BTN_2$= GetmostFrequ (NBT-Verb)
Return $BTN_2$-Verb [ ], Unknown-Verb [ ]
Def Find-VerbNet-Class (CS-Verb):
    For Verb in CS-Verb:
            Verb-Class=VerbNet.Classids(Verb. Strip ())
        If Verb-Class=! =[ ].
            For V in Verb-Class:
                Verb-VN-Category. Append (V)
Return Verb-VN-Category
```

Figure 3.6. VerbNet Technique Algorithm.

Figure 3.6 illustrates the algorithm used for the *VN* technique. As an input for the algorithm, it starts by reading the output verb lists from WordNet. It then checks unknown verbs in Bloom's list to return the verb class from the *VN* database. After it returns the verb class from the *VN* database, new verbs are added to the known verbs as part of Bloom's list. In case the verb class returns nothing for the verb, the algorithm uses the verb member to check the availability of having new verb members for the verb under study and checks if the new verb is in Bloom's list. If so, the verb level is returned.

If the verb is found with neither a class nor members in the *VN* database, the list is saved as unknown verbs in Bloom's Taxonomy. A limitation for *VN* includes gaps between verbs in the database; for that reason, some of the verbs are not found in the *VN* database. Finally, for those verbs whose classification is not found, the algorithm starts the classification process over for verbs but uses a different methodology (the *SVD* method, which will be explained in detail in the next section).

3.5 Singular Value Decomposition (*SVD*) Technique

In this section, verbs are classified based on Latent Semantic Analysis (*LSA*). *LSA* is a theory and method for extracting and representing the usage and meaning of domain concepts by using statistical computations (Golub & Reinsch, 1970). The process is divided into two tasks: calculating *SVD* to divide matrix A into three matrixes; and finding the verb level in Bloom's Taxonomy by applying *SVD* to the matrix (*A*). Doing this will break down each dimension in the matrix using Equation 3.1.

$$A_{v\times s} = U_{v\times v} \times S_{s\times v} \times {V^T}_{s\times s} \qquad (3.1)$$

Where:

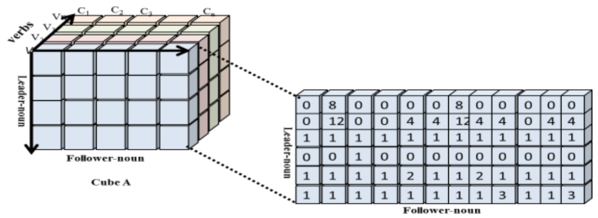
A*: v x s* matrix (*v* verbs, *s* sentences)

U*: v x v* matrix (*v* verbs, *v* verbs)

***V**: s x s* matrix (*s* sentences, *s* sentences)

```
Def Singular Value Decomposition (SVD) Technique ( ):
Input:  A= Matrix and BTN$_2$-Verb [ ]
Output: U matrix // Dimension Reduction Matrix
Def Calculate ():
    U, S, $V^T$ = SVD(A)  // U, S, $V^T$ matrixes
    UR=U [: 0:3]   //The dimensional Reduction of U
    VR=$V^T$ [0:3,]   //The dimensional Reduction of $V^T$
Def VerbClassify (VR):
    For all V ∈ VR Do
        Verb-Class, Unknown-Verb. ← Check-Class (V, BTN$_2$-Verb)
        V-Class ← ComputeNearstNighbor (V, Verb-Class, Unknown-Verb)
Return Verb-Class ()
Def Check-Class (V, BTN$_2$-Verb):
    If V in BTN$_2$-Verb ():
        BTN$_2$-Verb [Verb]=Level
        BTN$_2$-Verb [Verb]. Append (Level)
    Else:
        Unknown-Verb [Verb]. append (Zero)
Return Verb-Class [ ], Unknown-Verb [ ].
Def Compute-Nearest-Neighbor (V, Unknown-Verb, Verb-Class):
  //Find the K nearest neighbors for Unknown Verb based on the Euclidean distance
    For V in Unknown-Verb ():
    For $V_i$ in Verb-Class ():
    D ← Compute the distance d (V, $V_i$)
    Sort all D according to d (V, $V_i$)
    Select the first k points from D, those are the k closest distance.
    Select the verb and the class based on D
    Return Verb-Class ()
```

Figure 3.7. Singular Value Decomposition (SVD) Technique Algorithm



0	8	0	0	0	0	8	0	0	0	0	0
0	12	0	0	4	4	12	4	4	0	4	4
1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	0	0	0	0	0	0	0	0	0
1	1	1	1	2	1	1	2	1	1	1	1
1	1	1	1	1	1	1	1	3	1	1	3

$$U = \begin{bmatrix} -0.45 & 0.65 & -0.56 & 0.04 & -0.19 & 0.07 \\ -0.86 & -0.16 & 0.44 & -0.01 & 0.17 & -0.03 \\ -0.10 & -0.30 & -0.37 & -0.27 & -0.04 & -0.83 \\ -0.01 & -0.05 & -0.36 & -0.16 & 0.90 & 0.17 \\ -0.12 & -0.39 & -0.24 & -0.65 & -0.32 & 0.49 \\ -0.14 & -0.54 & -0.39 & 0.69 & -0.10 & 0.18 \end{bmatrix} \quad S = \begin{bmatrix} 22.54 & 0 & 0 & 0 & 0 & 0 \\ 0 & 7.07 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2.75 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2.17 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.96 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.55 \end{bmatrix}$$

$$V^T = \begin{bmatrix} -0.01 & -0.64 & -0.01 & -0.18 & -0.01 & -0.10 & -0.18 & -0.17 & -0.18 & -0.02 & -0.16 & -0.18 \\ -0.18 & -0.29 & -0.18 & -0.17 & -0.02 & -0.26 & -0.11 & -0.42 & -0.86 & -0.22 & -0.12 & -0.01 \\ -0.17 & -0.01 & -0.11 & -0.23 & -0.16 & -0.30 & -0.38 & -0.21 & -0.76 & -0.33 & -0.19 & -0.07 \\ -0.01 & -0.18 & -0.01 & -0.11 & -0.04 & -0.02 & -0.01 & -0.01 & -0.01 & -0.19 & -0.22 & -0.44 \\ -0.32 & -0.17 & -0.20 & -0.21 & -0.01 & -0.21 & -0.82 & -0.02 & -0.45 & -0.14 & -0.02 & -0.01 \\ -0.31 & -0.22 & -0.18 & -0.01 & -0.17 & -0.33 & -0.21 & -0.33 & -0.01 & -0.22 & -0.01 & -0.43 \end{bmatrix}$$

Figure 3.8. Singular Value Decomposition Matrixes.

As part of applying *SVD* (Wall, Rechtsteiner, & Rocha, 2003), dimensionality reduction techniques are utilized in order to reduce the high dimensionality of the Verbs matrix (U). Only 2-dimensions are considered here as illustrated in Figure 3.9. The biggest reason *SVD* is used to transfer this study's problem into a mathematical-based article is because it finds a reduced dimensional representation of the study's matrix that emphasizes the strongest relationships and that removes any noise.

The algorithm in Figure 3.7. represents the Singular Value Decomposition (*SVD*) technique. The algorithm uses the Checkclass function, and each verb in the verb list is checked to see if it belongs in Bloom's Taxonomy or not. If the verb is found in Bloom's list, the verb's level (BL_1, BL_2, BL_3, *and* BL_4) is returned as a verb class. Otherwise, it will return as not found (as in Table 3.1).

Next, verbs are classified by using a Nearest-Neighbor function (Jiang, Pang, Wu, & Kuang, 2012). This is done by computing the distance between two verbs after the two dimensions are extracted from the *U* matrix. Equation 3.2 is used to calculate Euclidean distance (*d*) between each of the two verbs.

$$V(d) = \sum_{i=1}^{n} (v_i - v_{i+1})^2 \qquad (3.2)$$

Table 3.1: Class label of Bloom's verbs.

Verb	**Returned-class**	**Dimensions from U matrix**
Use	BL$_1$	(-0.45,0.65)
Analyze	BL$_3$	(-0.86, -0.16)
Start	BL$_3$	(-10.-30)
Give	BL$_3$	(-0.01, -0.05)
Build	Not found	(-0.12, -0.39)
Repeat	Not found	(-0.14, -0.54)

It is necessary to compute distance between each two verbs' dimensions. These were normalized by scaling them between 0 and 1 (as seen in Table 3.2), and by using Equation 3.3.

$$d'_i = \frac{d_i - d_{min}}{d_{max} - d_{min}} \qquad (3.3)$$

The dimensions are scaled to fit into a specific range. Many types of normalization exist; this study used Min-Max Normalization. Min-Max Normalization transforms a value which fits in the range [0, 1] (as in Equation 3.3).

Table 3.2: Normalized Dimensions for Bloom's Verbs.

Verb	Dimensions from U matrix	Normalized dimension
Use	(-0.45,0.65)	(1,0.8)
Analyze	(-0.86, -0.16)	(0.6,1)
Start	(-10.-30)	(1,0.9)
Give	(-0.01, -0.05)	(1,0.7)
Build	(-0.12, -0.39)	(0.8,1)
Repeat	(-0.14, -0.54)	(0.4,1)

It is shown that the verb with the closest distance for Build is Use, and the verb with the closest distance for Repeat is Give.

Table 3.3: Distance between Bloom's Verbs.

	Use	**Analyze**	**Start**	**Give**
Build	0.09	0.30	0.14	0.77
Repeat	1.23	0.90	1.012	0.43

Finally, table 3.4 illustrates the cognitive skill class for each verb with the name code for the cognitive skill class.

Table 3.4: The Classified Bloom Verbs.

Verb	Use	Analyze	Start	Give	Build	Repeat
BT-class	BL_1	BL_3	BL_2	BL_2	BL_1	BL_2

3.6 Experiment Results and Evaluation

3.6.1 Experiment Results

This study tests the techniques using three high-quality textbooks found in Computer Science classes as course materials in many universities. The study's proposed techniques are then applied to see how they perform on these textbooks. Three text corpora are obtained for this task: "*Introduction to Algorithms*," "*Data Structures and Algorithms*," and "*Algorithms*," respectively. The experimental results and evaluations show that when performing the study's proposed task, the techniques were effective in classifying verbs based on the Cognitive Skill Levels. Sample of the CS-Verb classification provided in Appendix C.

This Chapter specifically focuses on classifying the extracted Computer Science action verbs based on their Cognitive Skill Levels. The verbs are used to describe the learning objectives.

As more details of the classification resulted (see Figure 3.10), a prominent feature is that significantly equal percentages of the verbs fell in BL_2*,* BL_3*, and* BL_4, while the percentage of the verbs is highest in BL_1. The results of applying this classification show that the textbook used to describe a low Cognitive Skill levels is found in the undergrad level; the learning objectives for this book will therefore be a prerequisite for the advanced courses of the algorithm. On the other hand, there are equal opportunities for high Cognitive Skill levels in the textbook.

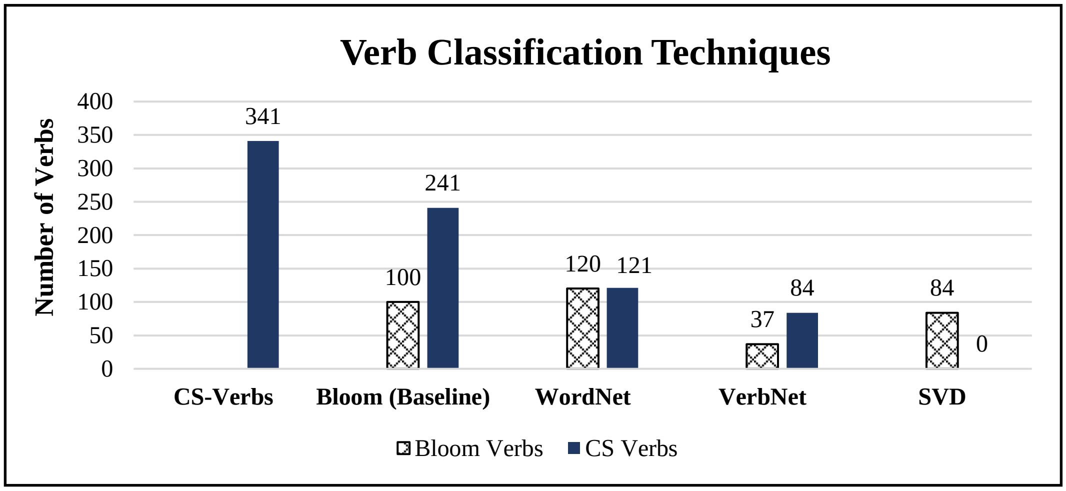


Figure 3.10. Verb Classification Based on the Three Techniques.

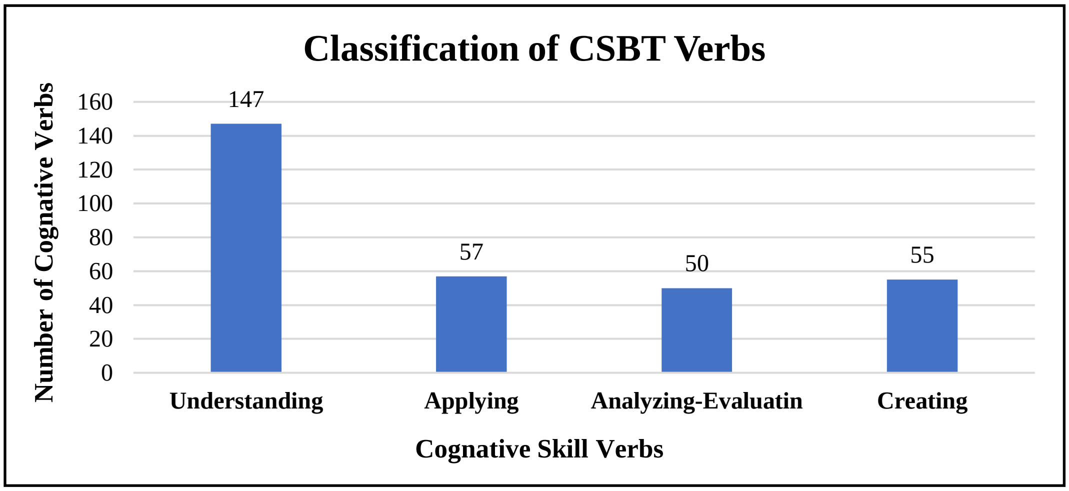


Figure 3.9. Verbs Classification based on the Cognitive Levels.

3.6.2 Evaluation Measures

As an evaluation step, the gold standard for any linguistic analysis is human judgment. In this chapter, statistical measures were used to estimate the agreement between the human classification of the verbs as well as the agreement between the results of verb classification and the “gold standard.” There are different measures of the agreement; for this, Fleiss' kappa measure was applied from the fields of inter-rater agreement (McHugh, 2012).

Fleiss' kappa *(£)* is a statistical measure for assessing the reliability of agreement between a fixed numbers of raters when assigning categorical ratings to classifying items. This measure calculates the degree of agreement in classification over that which would be expected by chance.

In this result, humans share intuitions about the analysis. For the techniques output, the classified verbs were given to native English speakers who are graduate students. This is typically done by checking to see if they agree or disagree with the automatic classification of the verbs. Apart from the cognitive validation of the analysis, the majority agreed that the verb classification could be used as a baseline classification for Computer Sciences to describe the learning objective.

3.7 Summary and Discussion

This chapter has described and discussed the concept of using Bloom’s Taxonomy in the field of computer science. Automatic techniques that are used to classify the verbs according to cognitive skill levels have been presented. The techniques are a sub-task of previous works (Nafa, Khan, Othman, & Babour, 2016a, 2016b).Classifying verbs based on cognitive skill levels is a novel and challenging problem.

The classification techniques make use of the cognitive domain in Computer Sciences. Not all the verbs found in the corpuses are equally important in the process of extracting the learning objectives; the most informative are the action verbs. These verbs are automatically classified using proposed techniques; Bloom suggested a short verb list to be used as a baseline. The techniques are also able to recover verbs that are relatively infrequent or specialized and thus unlikely to be captured manually by an expert. The results show that the classification of verbs overlaps between Cognitive Skill Dependencies; one verb can be in more than one level based on its function as a cognitive verb level. This adds a different nuance when describing the learning objectives. Based on the study's analytical result, it is possible to conclude that by using Cognitive Skill Dependencies, a teacher can decide which verbs to use at which level to match with the learner's skills, thus helping in writing the learning objectives. The final form of the output in this phase represented as a Semantic Knowledge Map (*SKM*) where the connection between concepts are cognitive Skill and WordNet relationships.

The next Chapter will start using Semantic Knowledge Map (*SKM*) as a graphical lexical source for building the Skill Inference Rules (*SIR*). It also answers a question raised here, which is how to logically design a schema as rule templates so that Cognitive Skill Dependencies and it is internal relationships can then be mapped to logic rules.

CHAPTER 4: CONSTRUCTING SKILL INFERENCE RULES (SIR)

4.1 Introduction

This chapter introduces an interesting approach that jointly models Semantic knowledge and First Order Logic (*FOL*). An *FOL* approach is adapted for generating Skill Inference Rules (*SIR*) from Semantic Knowledge Map (*SKM*). *SIR* is an interlingua that is used to translate the structure of *SKM* by emphasizing Cognitive Skill and WordNet (*WN*) relationships rather than the semantic relationships. Extending the *SKM* from a logic perspective increases its representational power and modelling capabilities to infer cognitive skill dependencies. Knowledge correlations in the *SKM* can be analyzed to infer *SIR* and to predict new facts. *SIR* describes how relations are associated in the *SKM*. The *SIR* involves the most complex mathematics in graph analysis, requiring intensive study to attain full comprehension. A number of researchers introduce the extraction of *FOL* from *SKM* (Gad-Elrab, Stepanova, Urbani, & Weikum, 2016; Guo, Wang, Wang, Wang, & Guo, 2016). However, each of them targets a very specific area, and their research has been designed to serve only their domains. The previous chapter presented the classification of cognitive CS verbs; this is used to identify both the internal relationship between concepts in *SKM* and other types of relationship between concepts (by using *WN* in *SKM*). This chapter makes use of those relationships (*Cognitive Skill* and *WordNet*) to generate *SIR*. This chapter also walks through an introduction of the knowledge representation *FOL* and goes into how to translate *SKM* to generate *SIR's*. Subsequently, the Chapter describes in detail each type of *SIR*. Finally, an explanation of *SIR* format is provided with language and graphic descriptions.

4.2 Logic as a Knowledge Representation Formalism

The main component of logic as a knowledge representation formalism is the knowledge base (KB) (Russell & Norvig, 2016). Each logical *KB* is composed by a set of rules stated in a logical language. These rules are expressed according to the syntax of the logical language. The main goal of logical *KB* is to infer new relationships from existing knowledge. For example, fatherOf (*x*, *y)* >> sonOf (*y*, *x*), in every *KB* if *x* is father of *y*, it is implied that *y* is also the son of *x*. In this example, the inference is defined as a technique to infer a new relationship from *KB*; if we know that *x* is the father of *y*, then *y* also being the son of *x* could be inferred easily. There are different proposed algorithms to infer the relationships in *KB*.

To generate simple or complex rules from *KB*, logical connectives (see Table 4.1) can be used. The basic representation for the *KB* is *FOL*, where the atomic formulas are predicates that assert a relationship among certain elements. In the previous example, fatherOf (*x*, *y*) is called an Atomic formula, where Complex formulas are computed recursively using truth Tables (as in Figure 4.1).

Truth tables are logic tables which list all possible values of the logical variables in any logical statement. Each logical variable can take only two values; a statement with n variables requires a table with 2^n rows. Truth tables are constructed from *IR* by transforming expressions into atomic formulas.

More explanation in the next section, like generating *SIR's* from *SKM*, is introduced.

Table 4.1. KB logical Connectives

Connective	**Meaning**
$\neg$	Not
$\wedge$	And
$\vee$	Or
$\Rightarrow$	Implication
$\Leftrightarrow$	Biconditional

Table 4.2. Truth Tables

A	B	$\neg A$	$A \wedge B$	$A \vee B$	$A \Rightarrow B$	$A \Leftrightarrow B$
False	False	True	False	True	True	True
False	True	True	False	True	True	False
True	False	False	False	True	False	False
True	True	False	True	False	True	True

4.3 Semantic Knowledge as Skill Inference Rules (*SIR*)

This section explains the First Order Logic (*FOL*) representation used to construct *SIR* from *SKM*. Translating *SKM* to *SIR* by using If-then rules are applied to *SKM*. The cognitive verb levels among concepts are extracted. Then, the lexical database is used to extract WordNet (*WN*) relationship among concepts obtained. The *SKM* represents the connection between concepts in form of WordNet and Cognitive relationships (Guo et al., 2016).

4.4 Skill Inference Rules (*SIR*) Mining

The representation of *SIR* is based on using If-then rules. If-then rules are applied to *SKM* to generate different templates of Skill Inference Rules. Based on the Cognitive Skills relationships that generated in Chapter Two *SIR* derived from it. The structure of *SIR's* used the formal structure of the First Order Logic (*FOL*). The *FOL* defines different types of symbols (Lyons, 1968):

Constant: It represents the domain objects.

Logic Variable: It stands for any one object in the domain (e.g. *x, y, z*) that is used to write the rules.

Predicate: It describes the relationships between two concepts. There are four relationships: *Understanding*, *Appling*, *Analyzing-Evaluating*, and *Creating*. It is important to note that each predicate has a value of either True or False. In the generated rules, it is necessary to decide whether the rules are True of False. This study makes the assumption that all of the generated rules are True.

This Chapter focuses primarily on mining *SIR* over predication. Cognitive relationships between each two concepts translates into multiple atoms. The conversion is applied to the graph so that it consists of a diverse collection of atoms. Four different rules are generated: Skill Inference Rules based on Cognitive predication, Skill Inference Rules based on Hypernym predication, Skill Inference Rules based on Hypernym and Cognitive predication, Skill Inference Rules based on Hyponym predication, and Skill Inference

Rules based on Hyponym and Cognitive predication. The next sections describe those rules in detail.

4.5 Skill Inference Rules (*SIR*) Based on Cognitive Prediction

These are rules of inferring the relationship between two nodes in *SKM* where there is no direct edge in between by using the directly connected relationships. For example, consider three nodes *A*, *B*, and *C*. If node *A* is connected to node *B* and node *B* connected to node *C*, then there is a heightened probability that node *A* will also be connected to node *C*. In the language of social network, the friend of your friend is also likely to be your friend. In terms of graph topology, transitivity means the presence of a heightened number of triangles in the graph. The Skill Inference Rules, with respect to cognitive connection are divided into four kinds: Skill Inference Rule (*Understanding*), Skill Inference Rule (*Analyzing*), Skill Inference Rule (*Appling*), and Skill Inference Rule (*Creating*). As shown in Figure.5.1, the Figure illustrates three formats *A*, *B*, and *C*, where *A* is the language description of each *SIR*, *B* is the graphic illustration, and *C* is the *SIR* template that is used in the next chapter to infer Cognitive Skill Dependencies (*CSD*) between concepts.

Skill Inference Rules (*Understanding*) is a connection between two concepts that uses the Understanding cognitive relationship as a pivotal part of the rule. Different combinations of the Skill Inference Understanding based with *WN* relationships also exist (as illustrated in Table 5.3).

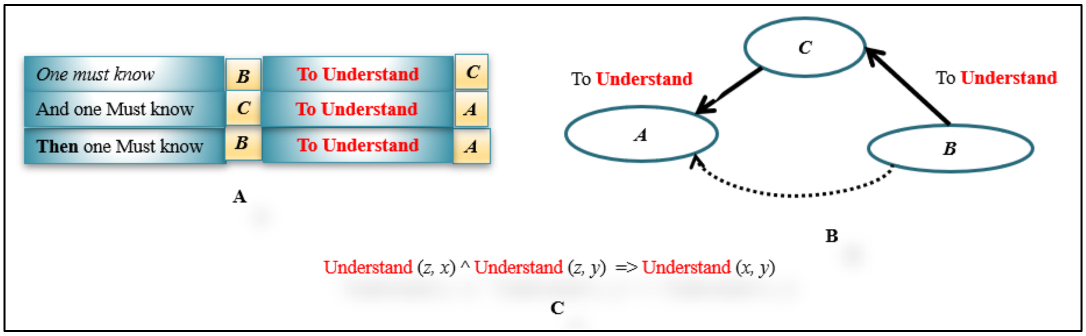


Figure 4.1. Skill Inference Rule Understanding.

Table 4.3. Combination of the Skill Inference Rule Understanding with WN.

Rule Name	IF (A and B)	THEN(C)
Understanding. Understanding	**$BL_1. BL_1$**	**BL_1**
Understanding. Applying	$BL_1. BL_2$	$BL1\|BL_2$
Understanding. Analyzing	$BL_1. BL_3$	$BL1\|BL3$
Understanding. Creating	$BL_1. BL_4$	$BL_1\|BL_4$
Understanding. Superclass	$BL_1. W_1$	$BL_1\|W_1$
Understanding. Subclass	$BL_1. W_2$	$BL_1\|W_2$
Understanding. Super part	$BL_1. W_3$	$BL1\|W_3$
Understanding. Subpart	$BL_1. W_4$	$BL1\|W_4$

Skill Inference Rule (*Applying*) is a Skill Inference Rule that uses *Applying* relationships as a pivotal part of the rule. Different combinations of the Skill Inference Applying based rule exist (as illustrated in Table 5.3).

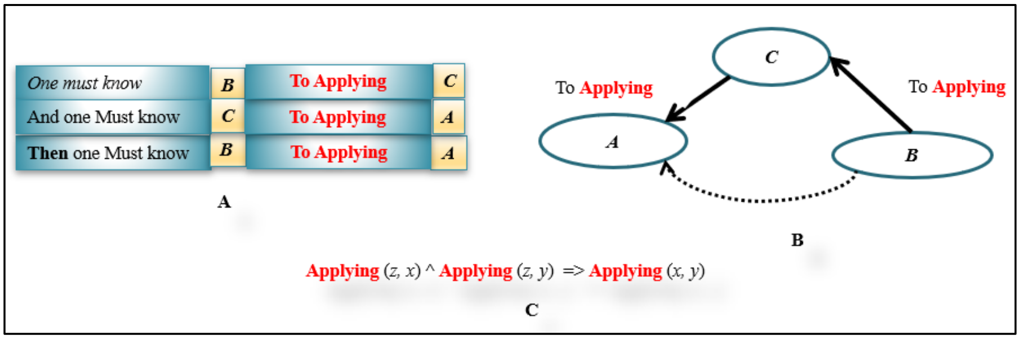


Figure 4.2. Skill Inference Rule Applying.

Table 4.4. Combination of the Skill Inference Rule Applying with WND Rules.

Rule Name	IF (A and B)	THEN(C)
Applying. Applying	***BL2. BL2***	***BL2***
Applying. Understanding	*BL2. BL*$_1$	*BL2|BL1*
Applying. Analyzing	*BL2. BL3*	*BL2|BL3*
Applying. Creating	*BL2. BL4*	*BL2|BL4*
Applying. Superclass	*BL2. W1*	*BL2|W1*
Applying. Subclass	*BL2. W2*	*BL2|W2*
Applying. Super part	*BL2. W3*	*BL2|W3*
Applying. Subpart	*BL2. W4*	*BL2|W4*

Skill Inference Rule (*Analyzing*) is cognitive level that uses analyzing relationships as a pivotal part of the rule. Different combinations of the Skill Inference *Analyzing* rule also exist (as illustrated in Table 5.5).

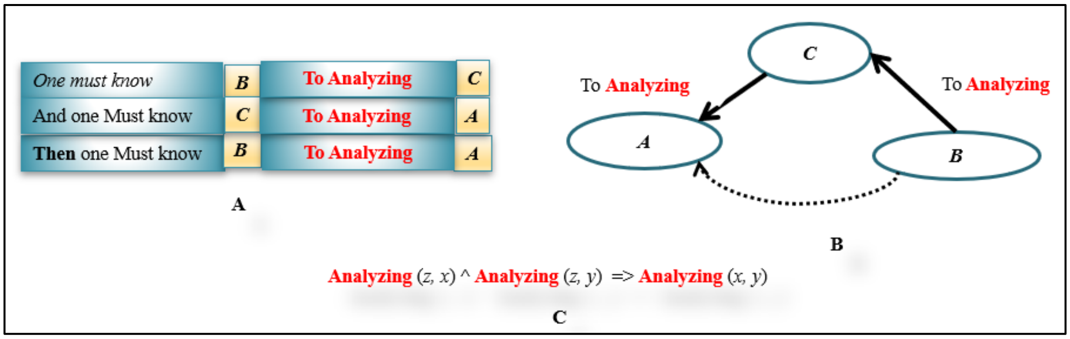


Figure 4.3. Skill Inference Rule Analyzing.

Table 4.5. Combination of the Skill Inference Rule Analyzing with WND Rules.

Rule Name	IF (*A and B)*	THEN(*C*)
Analyzing. Analyzing	***BL3. BL3***	***B3***
Analyzing. Understanding	$BL3.\ B_1$	*BL3|BL1*
Analyzing. Applying	$BL3.\ BL_2$	*BL3|BL2*
Analyzing. Creating	*BL3. BL4*	*BL3|BL4*
Analyzing. Superclass	*BL3. W1*	*BL3|W1*
Analyzing. Subclass	*BL3. W2*	*BL3|W2*
Analyzing. Super part	*BL3. W3*	*BL3|W3*
Analyzing. Subpart	*BL3. W4*	*BL3|W4*

Skill Inference Rule (*Creating*) is a cognitive level that uses *Creating* cognitive relationships as a pivotal part of the rule (as illustrated in Figure 4.4). Different combinations of the Skill Inference Creating (as illustrated in Table 4.6).

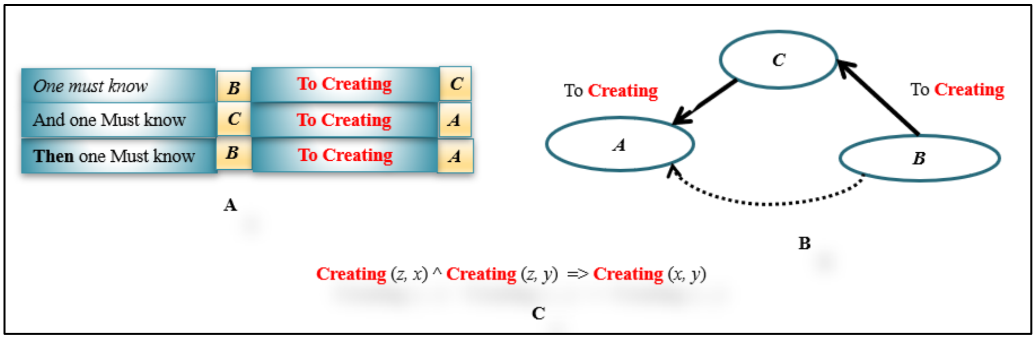


Figure 4.4. Skill Inference Rule Creating

Table 4.6. Combination of the Skill Inference Creating with WND.

Rule Name	IF (*A and B*)	THEN(*C*)
Creating. Creating	***BL4. BL4***	***BL4***
Creating. Understanding	$BL4.\ BL_1$	*BL4*|*B1*
Creating. Applying	$BL4.\ BL_2$	*BL4*|*B2*
Creating. Analyzing	$BL4.\ BL_3$	*BL4*|*B3*
Creating. Superclass	*BL4. W1*	*BL4*|*W1*
Creating. Subclass	*BL4. W2*	*BL4*|*W2*
Creating. Super part	*BL4. W3*	*BL4*|*W3*
Creating. Subpart	*BL4. W4*	*BL4*|*W4*

4.6 Skill Inference Rules (*SIR*) Based on Hypernym Prediction

This type of rule is a rule of inferring the cognitive relationship using identified hypernym relationships among concepts. *A* hypernym is a particular semantic *WN* relationship represented as hierarchical relationships among concepts. *A* concept is considered a hypernym of another if its meaning lists the second concept as an example of its own meaning, for example, consider two nodes *A* and *B* and *A* has children A_i. If node *A* is connected to node *B* as a hypernym relationship, then there is a heightened probability that children of node *A* will also be connected to node *B* as a cognitive relationship. The Transitive relationship with respect to cognitive skill levels is divided into four kinds: *Understanding, Analyzing, Appling, and Creating* (As shown in Figure.4.5, Figure.4.6, Figure.4.7, and Figure.4.8, respectively).

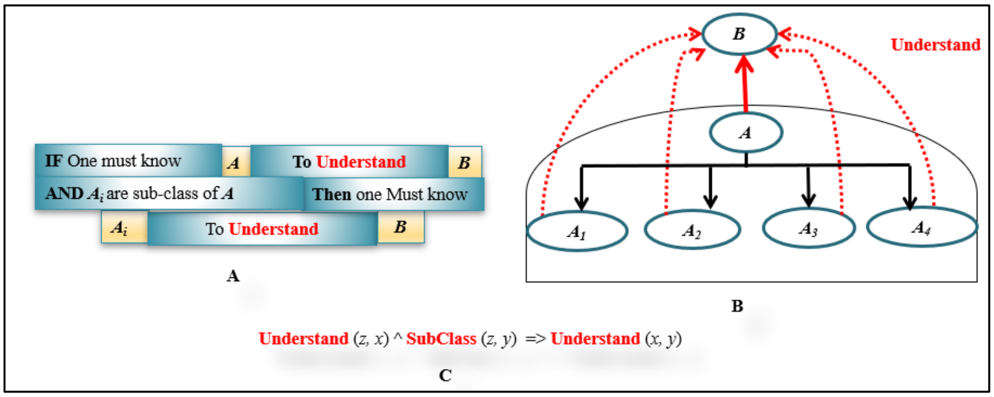


Figure 4.5. Skill Inference Rule Understanding and SubClass.

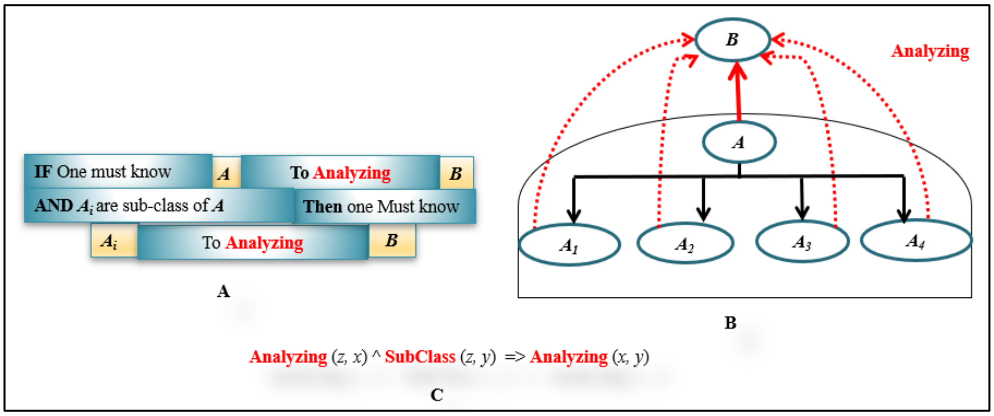


Figure 4.6. Skill Inference Rule Analyzing and SubClass

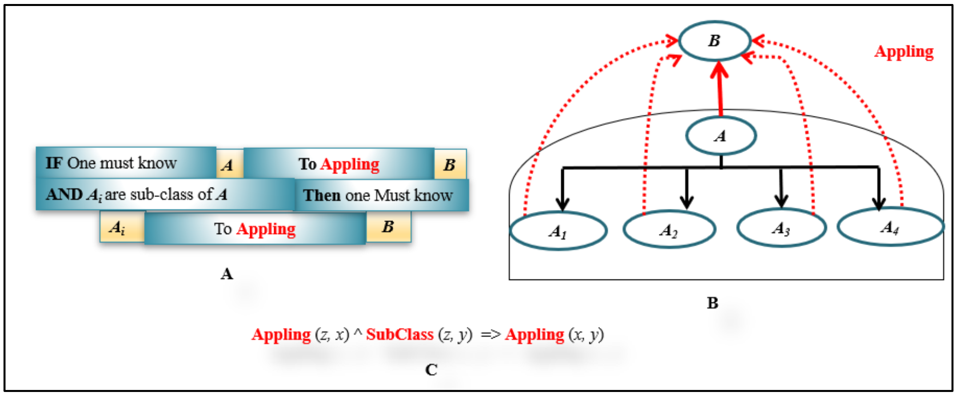


Figure 4.7. Skill Inference Rule Appling and SubClass

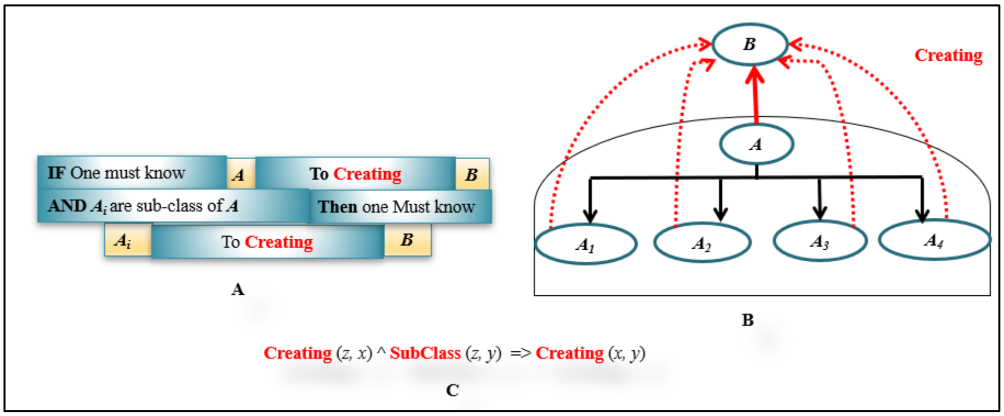


Figure 4.8. Skill Inference Rule Creating and SubClass.

4.7 Skill Inference Rules (*SIR*) Based on Hypernym and Cognitive Predication

It is a rule of inferring the Hypernym WordNet relationship using identified Cognitive relationships among concepts. For example, consider two nodes *A* and *B* and *A* has children A_i. If the children of concept *A* are connected to concept *B* in a Cognitive relationship, then there is a heightened probability that *A* will also be connected to node *B* as a Hypernym relations. *B* is then a hypernym of *A* if every *A* is a kind of *B*. In the language of social network, the friend of your friend is also likely to be your friend. In terms of graph topology, transitivity means the presence of a heightened number of triangles in the graph.

This rule with respect to cognitive skill levels is divided into four kinds: *Understanding, Analyzing, Appling, and Creating* (as shown in Figure.4.9, Figure.4.10, Figure.4.11, and Figure.4. 12, respectively).

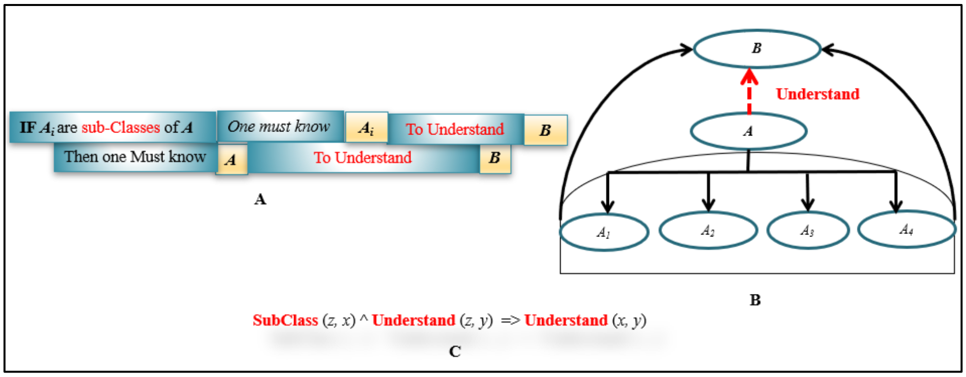


Figure 4.9. Skill Inference Rule SubClass and Understanding.

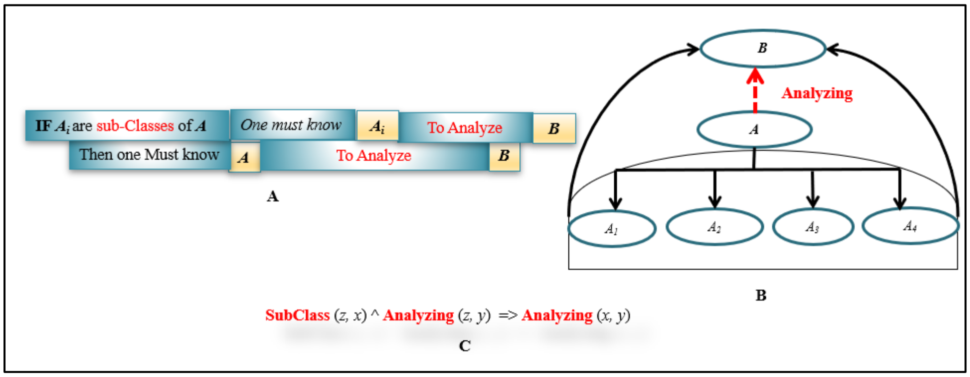


Figure 4.10. Skill Inference Rule SubClass and Analyzing.

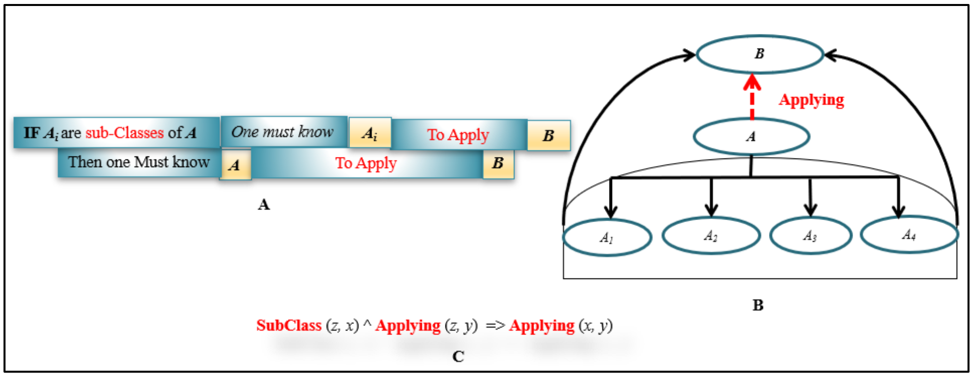


Figure 4.11. Skill Inference Rule SubClass and Applying

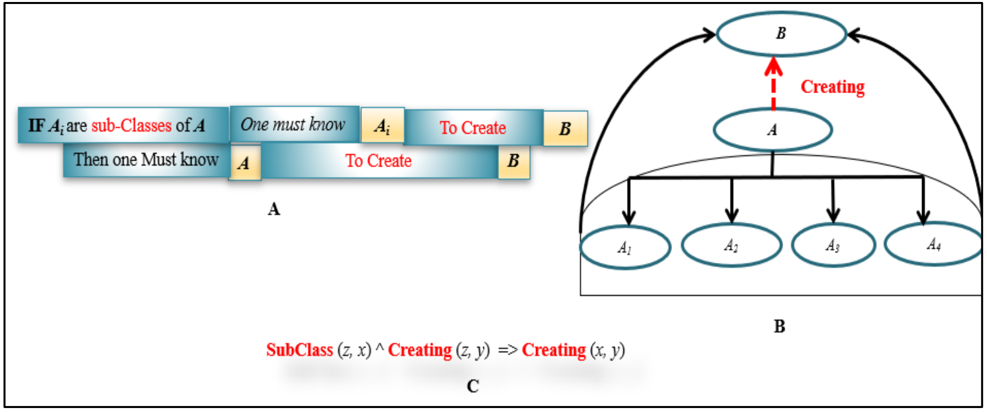


Figure 4.12. Skill Inference Rule SubClass and Creating

4.8 Skill Inference Rules (*SIR*) Based on Hyponym Predication

The opposite relationship for a hypernym in *WN* is a hyponym. This rule that concludes the cognitive relationship uses identified Hyponym relationships among concepts. For example, if two concepts *A* and *B* and *A* has children A_i, if node *A* is connected to node *B* as a Hyponym relationship, then there is a heightened probability that the children of node *A* will also be connected to node *B* as a *CSBT* relationship. *R* is then

Transitive if for all *A*, *B*, and *C*. In terms of graph topology, transitivity means the presence of a heightened number of triangles in the graph.

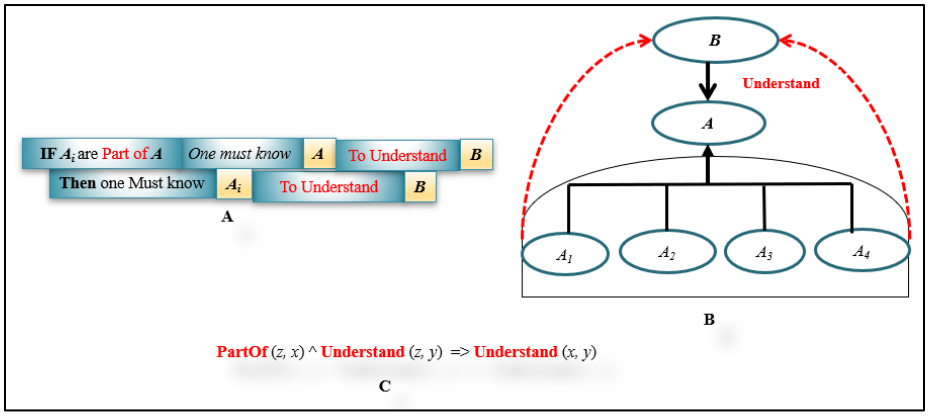


Figure 4.13. Skill Inference Rule Hyponym and Understanding

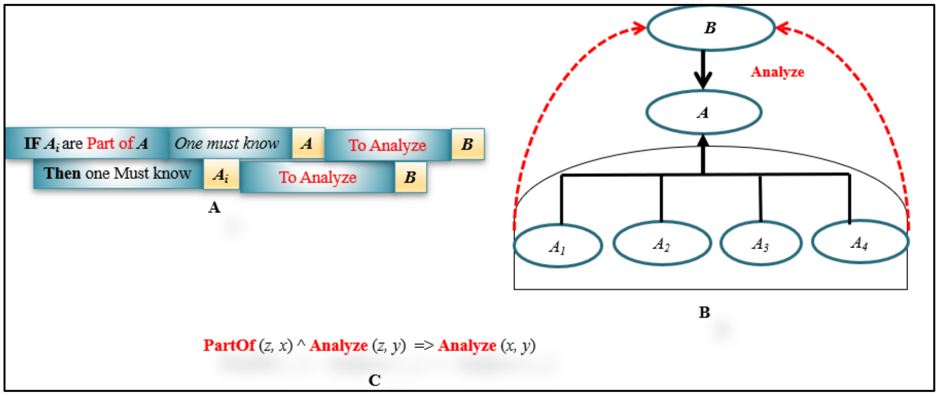


Figure 4.14. Skill Inference Rule Hyponym and Analyzing

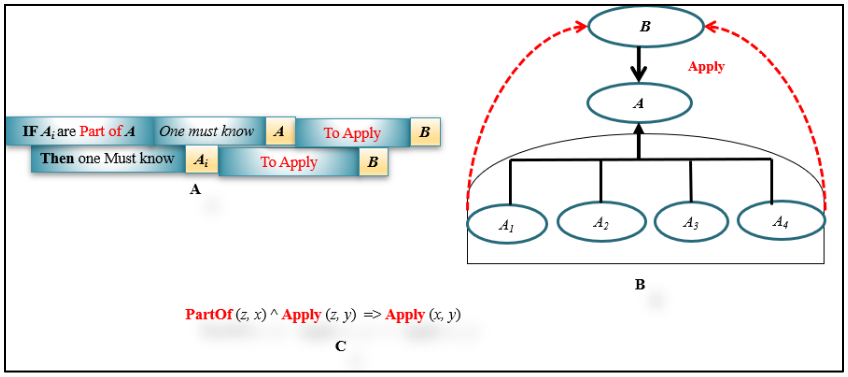


Figure 4.15. Skill Inference Rule Hyponym and Applying.

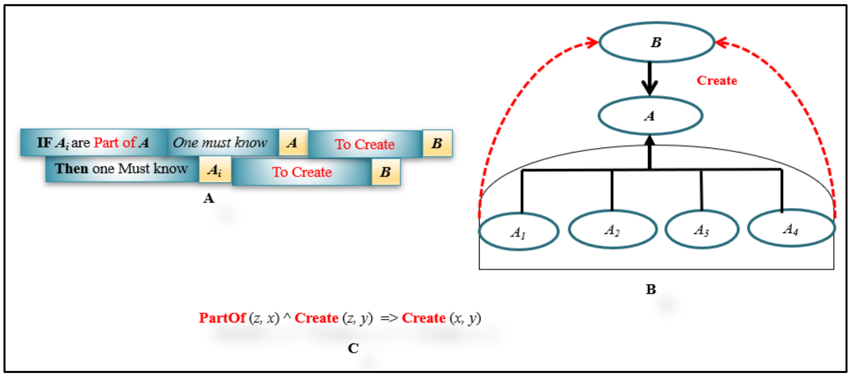


Figure 4.16. Skill Inference Rule Hyponym and Creating.

4.9 Skill Inference Rules (*SIR*) Based on Hyponym and Cognitive Predication

This rule infers the Hyponym relationship by using identified cognitive relationships among concepts. For example, if two nodes *A* and *B*, and *A* has children A_i if the children of concept *A* are connected to concept *B* as a cognitive relationship, then there is a heightened probability that *A* will also be connected to node *B* as a Hyponym relationship. *B* is a Hyponym of *A* if every *A* is a kind of *B*.

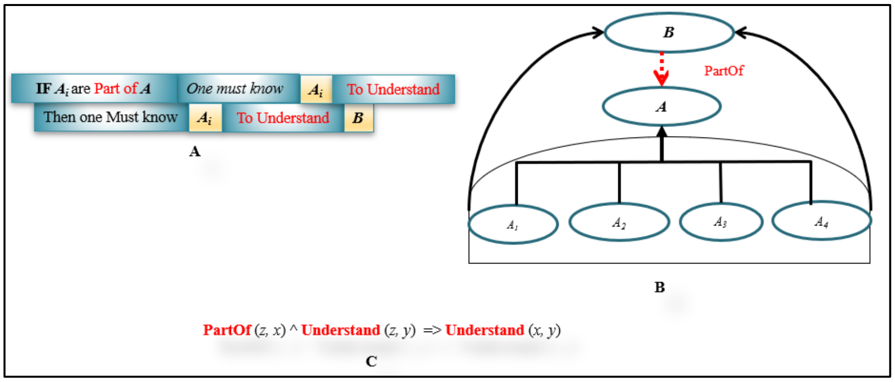


Figure 4.17. Skill Inference Rule Understanding and Hyponym.

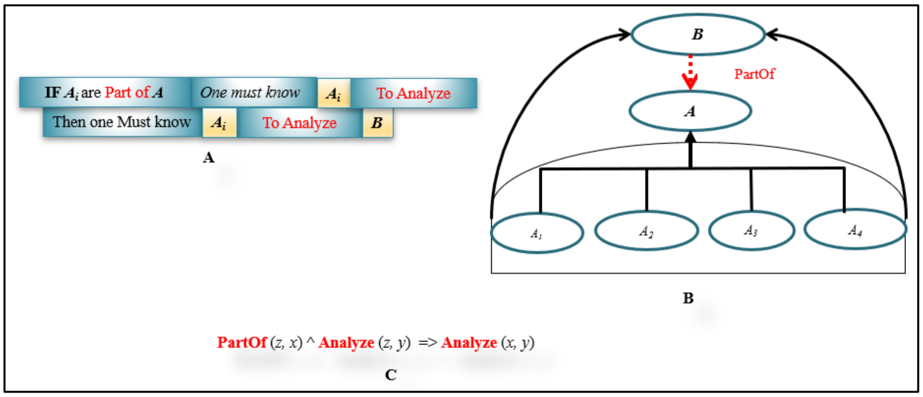


Figure 4.18. Skill Inference Rule Analyzing and Hyponym.

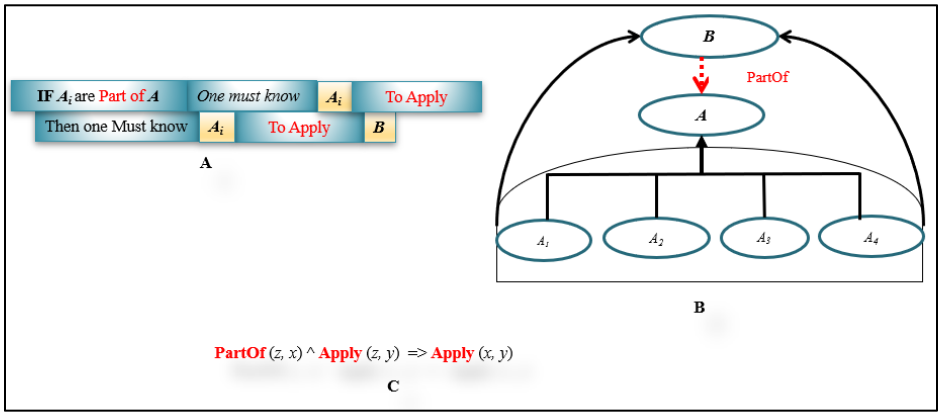


Figure 4.19. Skill Inference Rule Applying and Hyponym.

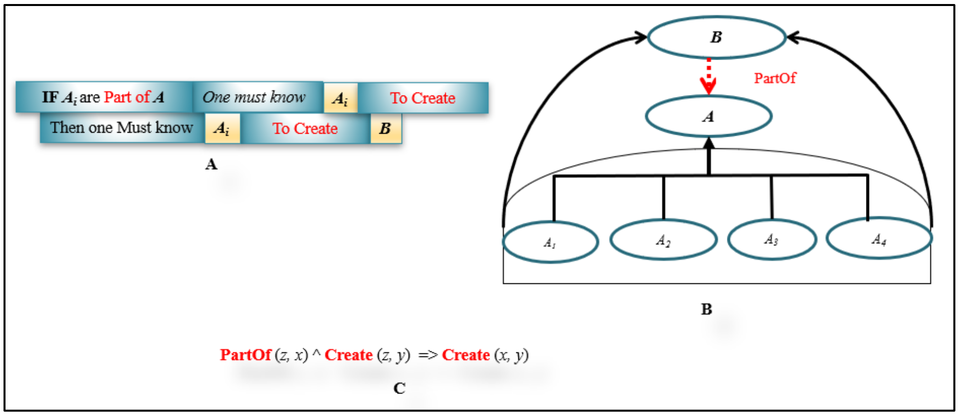


Figure 4.20. Skill Inference Rule Creating and Hyponym.

4.10 Summary and Discussion

This chapter implies *FOL* to facilitate the representation of *SKM* formalism to extract the *SIR's*. A novel and interesting logic foundation is used to produce a variety of *SIR's* templates from Cognitive Dependencies and WordNet Dependencies.

The Skill Inference Rules (*SIR*) were generated in this Chapter. The generated *SIR's* are simple because if the *SIR* is too complex, it is possible for it to be neither valid nor not-valid. In other words, there would be a chance of generating a not-valid *SIR*. The *SIR's* in this experiment are generated by a group of Ph.D. students in the research phase with expertise in the Algorithm area.

These templates are then used to infer Cognitive Skill Dependencies in the Chapter Six. Developing a straightforward and easy-to-implement methodology for transforming a *SKM* into the corresponding *SIR's* breaks many limitations and obstacles in the extraction of Cognitive Skill Dependencies.

The chapter demonstrates a background about knowledge representation and *FOL*. It also explains translating *SKM* to *SIR* and how to generate Cognitive Dependencies and WordNet Dependencies templates. The construction of the rules introduced by details via language and graphical description are different categories of templates that are proposed and constructed.

The next chapter introduces the proposed model to infer the Cognitive Skill Dependencies (*CSD*). It uses a probability-based inference. It will be described in greater detail in Chapter Five.

CHAPTER 5: THE APPLICATION OF MCKSN MODEL

5.1 Introduction

This final chapter builds upon the material discussed in the previous chapters. It explores the Markov Cognitive Logic State Network (*MCKSN*), a probability-based model that is used for inferring Cognitive Skill Dependencies attached with a degree of probability. In this chapter, the main concepts used in the *MCKSN* model are presented. First, the Markov Network is introduced as an essential concept for this study, followed by an explanation of Markov Logic Network (*MLN*), finally, a description of applying *MCKSN* model to infer Cognitive Skill Dependencies is presented.

5.2 Markov Network (*MN*)

The theory of the Markov Network (*MN*) provides suitable framework to model the dependencies between objects in a domain such as the dependencies between pixels in image processing tasks, identifying Twitter spammers based on their dependencies, and the analysis of social network structure and other interesting application. In this context, *MN* is used as a framework to model the dependencies between Cognitive Skill in graph form with respect to the node neighbors (set of cliques)

A Markov Network is defined as an undirected graph $MN= (V, E)$, where each node v_i in the node set V represents a random variable. $E \subseteq V\times V$ is a set of edges connecting the nodes. Each edge $e_{i,j}$ represents conditional dependence relationships between the random variables v_i and v_j. Two random variables are conditionally dependent on each other if they have an edge (direct link)(Getoor & Taskar, 2007; Wasserman & Pattison, 1996) .According to (Clifford, 1990; Kemeny, Snell, & Knapp, 2012; Li, 2009; Winkler, 2012) nodes in an *MN* represent random variables and Markovian properties described as the following:

- A node is conditionally independent of all other nodes, given its neighbors (as in Figure 5.1).

$$A \perp rest |\, B, C, D$$

- Any two non-adjacent nodes are conditionally independent of each other, given all other nodes (as in Figure 5.1).

$$A \perp G |\, rest$$

- Any two subsets of nodes are conditionally independent, given a separating subset where every path from a node going in the first subset to a node in other subset passes through the rest of the nodes (as in Figure 5.1).

$$A, B \perp F, G |\, C, D, E$$

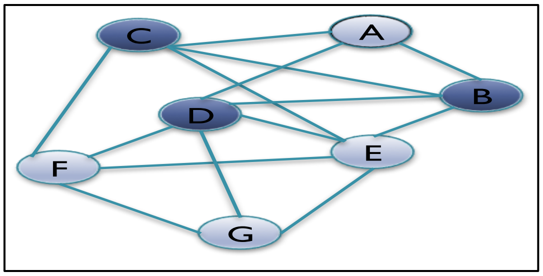


Figure 5.1. an Example of a Small Markovian Properties Graph.

The edges represent a direct probabilistic dependency between the random variables. If there is a direct edge between any two random variables, then there is a dependency connection between them. In *MN*, if two random variables don't have a dependency, they should not have a path. However, it is possible for there to be an indirect path, or for the variables to be indirectly dependent on each other.

The dependency connection between random variables could be a direct or indirect connection. Those types of connections are illustrated in Figure 5.2 (a) and5.2 (b).

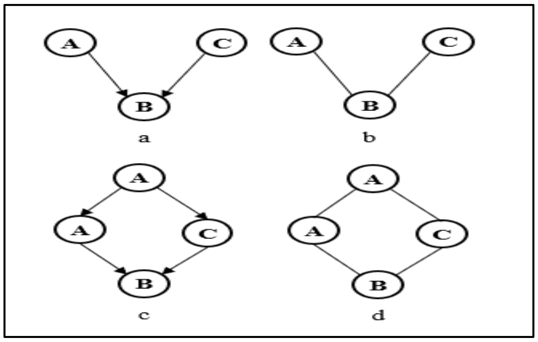


Figure 5.2. Example of the Nodes Connection in Markov Network.

Consider four nodes *A, B, C* and *D*. From a graphical point of view, the connection between nodes expresses the type of relationships between them if it is direct (Figure 5.2.a) or indirect (Figure 5.2.b). Suppose that the conditional probability represents the connection between nodes.

In Figure, 5.2.a the nodes *A* and *C* are independent of each other, but they are conditionally dependent given B. While Figure 5.2.b represents both situations incorrectly; the nodes *A* and *C* are dependent by transitivity but are conditionally independent given *B* (Sutton & McCallum, 2006).

This can be expressed in a slightly different way by considering the joint distribution of *A* and *B*, given *C*, which can be written as follows:

$$P(a,b|c) = p(a|c)p(b|c)$$

Figure 6.2.c shows a cyclic dependence structure represented as a directed graph where nodes are conditionally independent given their neighbors. While Figure 5.2.d captures the conditional independence of A and C given B and D, also, B and D are conditionally dependent given C. The modeling the conditional probability between random variables depends on the graph structure and direction (Friedman, 2004). The next part explains the input and output for the *MN* with a practical example.

5.2.1 Markov Blanket

The Markov Blanket of a random variable (Target) consists of all other random variables that make this target conditionally independent of all the other random variables. In other words, the node is independent of the rest of the nodes in the graph given its first level neighbors (Margaritis & Thrun, 2000).

Markov Network can answer many questions. One of the interesting question *MN* can answer is to estimate the probability of a given outcome of the random variable given the outcome of certain other random variables. The following section presented an example of using *MN* to estimate the hidden dependencies of the transmission of bad habitats between friends given some evidence in small social network.

5.2.2 Example of Markov Network

Assume that four random variables *A, B, C, D* (as in Figure 5.3) correspond with four students *Alice, Bob, Charles, Debbie* studying together in a group (Ivanova, 2017; Koller & Friedman, 2009).

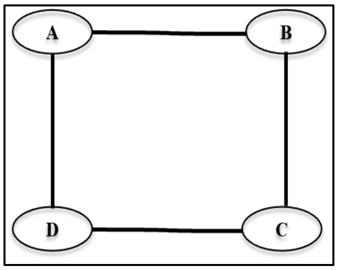


Figure 5.3. A Simple Example of a Markov Network

The potential function in this example indicates whether the students had a misconception due to the potentially confusing material (as given in Table 5.1). In this example, assuming that the query is the probability of Bob (*B*) having a misconception, given the evidence that Charles (*C*) does not have the misunderstanding, the probability is written as:

$$p(b_1|c_0)$$

As in Table 5.1, the numbers in each table indicate the local agreement of each variable takes a joint assignment. The intuition in this example says that if two friends (*A* and *B*) study together, then they will influence each other somewhat. As for this influence concerning misconceptions, they are represented in the formula as follows: a_0=has misconception a_1=no misconception b_0=has misconception, and b_1=no misconception. This indicates that if *A* and *B* are friends and study together then an edge between them indicates that if one of them has a misunderstanding, the other one is also likely to have the same misconception, denoted as (a_0,b_0), which is 30 in Table 5.1 Likewise, if one of them has no misconception, the others are also likely to have no misconception, denoted as (a_1,b_1) which is equal to 10 in Table 5.1. Finally, the other two numbers in the middle mean that their disagreement is very low. The table also shows the similar notions of happiness for the other pairs in the graph, given as *$\emptyset(B,C)$, $\emptyset(C,D)$, and $\emptyset(D,A)$.* Given the information from the graph, it is clear that *B* and *C* really seem to agree with each other. It's very difficult for them to have opposing opinions. This doesn't fit neatly into a directed graph, and because the influence flows in both directions, utilizing the Markov Network is proposed. On the other hand, *C* and *D* like to argue with each other all the time. For example, if one of them says that it's going to rain today, the other one is going to say that it's sunny today. Therefore, the preferred assignments for their local opinion would be the one in which they disagree with one another.

Table 5.1 A Simple Example of Markov Network Variables with their Potential Function Assignment.

Random Variables		$\emptyset(\boldsymbol{A},\boldsymbol{B})$
a_0	b_0	30
a_0	b_1	5
a_1	b_0	1
a_1	b_1	10

Random Variables		$\emptyset(\boldsymbol{B},\boldsymbol{C})$
b_0	c_0	100
b_0	c_1	1
b_1	c_0	1
b_1	c_1	100

Random Variables		$\emptyset(\boldsymbol{C},\boldsymbol{D})$
c_0	d_0	1
c_0	d_1	100
c_1	d_0	100
c_1	d_1	1

Random Variables		$\emptyset(\boldsymbol{D},\boldsymbol{A})$
d_0	a_0	100
d_0	a_1	1
d_1	a_0	1
d_1	a_1	100

Mathematically speaking, to define a joint probability distribution, the notion of the product of factors are used (Pearl, 2014). They can then be multiplied together using the formula as follows:

$$p(x) = \frac{1}{Z}\left(\prod\nolimits_{i=1} \emptyset_i(X)\right) \tag{5.1}$$

Where:

$\boldsymbol{X} \in \{A, B, C, D\}$

$\boldsymbol{x_1}$: student has the misconception

$\boldsymbol{x_0}$: Student does not have a misconception.

$\boldsymbol{Z}$**:** is a Partition Function.

$$Z = \sum_{X} \emptyset_i(x) \tag{5.1.1}$$

Applying Equation 5.1 to these factors resulted in the outcome illustrated in Table 5.2.

Table 5.2 A Simple Example of Normalized and Non-normalized Random Variables.

Random Variables					
A	*B*	*C*	*D*	Nonnormalized	Normalized
a_0	b_0	c_0	d_0	300000	0.04
a_0	b_0	c_0	d_1	300000	0.04
a_0	b_0	c_1	d_0	300000	0.04
a_0	b_0	c_1	d_1	30	0.0000041
a_0	b_1	c_0	d_0	500	0.000069
a_0	b_1	c_0	d_1	500	0.000069
a_0	b_1	c_1	d_0	5000000	0.69
a_0	b_1	c_1	d_1	500	0.000069
a_1	b_0	c_0	d_0	100	0.000014

a_1	b_0	c_0	d_1	1000000	0.14
a_1	b_0	c_1	d_0	100	0.000014
a_1	b_0	c_1	d_1	100	0.000014
a_1	b_1	c_0	d_0	10	0.0000014
a_1	b_1	c_0	d_1	100000	0.014
a_1	b_1	c_1	d_0	100000	0.014
a_1	b_1	c_1	d_1	100000	0.014
Total				7,201,840	

To normalize those factors, a partition function (Pearl, 2014) was used. This function can be seen as a normalizing constant that sums up all these entries, resulting in the value *Z* (as in Table 5.2), where *Z* = 7,201,840. By dividing all the entries by *Z*, the probability distribution is normalized (as in Table 5.2). Any desired probability can be obtained from the joint distribution as usual. This example attempts to compute the following:

$$p(b_1|c_0) = \frac{p(b_1 \cap\ c_0)}{p(c_0)} \tag{5.2}$$

$$p(b_1|c_0) = \tag{5.2.1}$$

$$\frac{0.000069 + 0.000069 + 0.0000014 + 0.014}{0.04 + 0.04 + 0.000069 + 0.000069 + 0.000014 + 0.14 + 0.0000014 + 0.014}$$

$$p(b_1|c_0) = \frac{0.014}{0.234} = 0.06 \tag{5.2.3}$$

Based on the results of this, it can then be concluded that if *Charles* does not have a misconception, *Bob* is only 6% likely to have one as well.

5.3 Markov Logic Network (*MLN*)

Markov Logic Network (*MLN*) provide a powerful probabilistic modeling framework based on first-order logic and probability inference to infer the hidden relationships between objects. *MLN* can be used in different application such extracting many kinds of syntactic and semantic information, Drug event extraction, and Concept extraction in ontology learning. There is different type of probability-based techniques that has been used in the previous studies for extracting the semantic relations from the text. But *MLN* add a new flavor in terms of using a descriptive language such as First Order Logic(*FOL*) to build rules that can guide the inference of the relationships.

The simplest way to understand Markov Logic as introduced by (Richardson & Domingos, 2006) is to combine *FOL* and *MN*. The main idea behind Markov Logic is that, by attaching strengths to a Rules template, these *IR's* can be used to infer fact between objects. The formal definition is mentioned in (Richardson & Domingos, 2006). *MLN* is formally defined as:

Definition: A Markov Logic Network is defined by *MLN* as an undirected graph $G= (V, E)$, where V is a set of facts, each node of which represents a fact. $E \subseteq V\times V$ is a set of edges connecting the facts and representing the correlations between the facts.

The nodes in an *MLN* represent facts, where the facts are equivalent to the random variables in *MN*. The fact describes relations between two logic variables which is the constants in the domain- Friends (‘*Anna*’, ‘*Bob*’). This type of fact is called a ground fact if it maps to specific individual objects in the domain. The set of ground facts define the structure of the Markov Logic Network (Mario&Matr, 2014). The facts in the real world is true or false in most domains it is challenging to come up with a fact that is always true, hence the reason for modeling random variables to facts. It is important to note that each fact has a value of either True or False. In the generated facts, it is necessary to decide whether the IR's are True or False.

The facts in an *MLN* should have the same assumptions as those in an *MN*. More than one fact can represent an *IR*. The *IR's* are expressed according to the syntax of the logical language. The primary goal of *IR's* is to infer new facts from existing ones. One example is fatherOf (*x*, *y*) >> sonOf (*y*, *x*); in every *IR*, if *x* is the father of *y*, it is implied that *y* is also the son of *x*. In this example, the inference is defined as a technique to infer a new fact from the *IR*. If we know that *x* is the father of *y*, then *y* also being the son of *x* could be readily inferred. For definitions and illustrative examples of the logic theories, the reader is invited to consult a textbook like (Fitting, 2012; Makkai & Reyes, 2006; Smullyan, 2012; Sowa, 2000). Two facts are connected by an edge if both facts appear in the same *IR* where the edge represents the dependency relationship between them. If the IR combines the relationship between a few facts, one may influence the other's dependence.

Input*:*

1. **Constants:** are fixed *objects* in the domain.

2. Hypothesis **Facts**: are an atomic formula consisting of a predicate with a suitable number of arguments. Each fact maps term to term where the terms is an expression representing an *object* in the domain.
3. Hypothesis **Inference Rule (*IR*)**: are *FOL*, *IR's* consisting of one *fact* or more connected using logical connectives. In the following example, facts are given about friendship relations and smoking habits.
4. The **Potential Function** is a composed of simple values for the *IR*, expressed with integers {0, 1}. For each *IR*, this function assumes a value of 1 for a state of the clique if the truth values of the nodes make the *IR* true, while considering a value of 0 for a state of the clique if the truth values of the nodes make the *IR* false.
5. **User Query**: estimates the probability of a given outcome of a fact given the outcome of certain other ground facts.

Output:

MLN can estimate the probability of a given outcome of a fact given the outcome of certain other ground facts.

5.4 Example of Markov Logic Network (*MLN*)

The motivating example is used to explain the procedure of applying *MLN*. This example will model the smoking habits between friends in a social network (Mario&Matr, 2014; Richardson & Domingos, 2006). In other words, people with friends who smoke would also smoke, and that people who smoke would then have cancer.

Based on several studies done between 1971-2000 about the smoking habits of a social network, the study showed that some people who don’t smoke but who have friends that smoke became smokers after several years. Also, people don’t stop smoking at random but stop smoking in clumps. If a friend of a person stops smoking, that means that the person will be more likely to quit smoking themselves, and if the friend doesn’t stop, the person would have a harder time to stop smoking by themselves. Using *MLN*, this problem can be modeled by using a few facts and *IR’s*. To introduce the input for the *MLN* model by using the example above, suppose that the *IR’s* in Figure 5.4 and Figure 5.5 are given. The *IR* in Figure 5.4 means that smoking causes cancer. Also, the *SIR* in Figure 5.5 implies that IF *x* and *y* are friends and *x* is a smoker that means that *y* will be a smoker too due to influences from each other. The input for the *MLN* procedure is as follows:

- **Constant:** assume that there is a very simple social network which includes two people (‘**Anna’** and ‘**Bob’**).
- Two facts

Smokes (*x*) => Cancer (*x*)

Figure 5.4. The Fact Template

Friends (*x, y*) => (smokes (*x*) <=> smokes (*y*))

Figure 5.5. The Inference Rule (IR) Template

- A **fact** about this social network (Smokes(‘*Bob’*)).
- The **potential function** is defined as:

$$f(x) = \begin{cases} 1 & \text{if} \quad IR \text{ is true(Has value} = 1 \text{ in the truth table)} \\ 0 & \text{otherwise} \end{cases}$$

- **The query** is that if there is the fact that *Bob* smokes and he is *Anna's* friend and *Anna* does not smoke, what is the probability that *Anna* will become a smoker, and that both *Anna* and *Bob* will develop cancer.

```
Procedure 1:  Markov Logic Network
Input:  The Inference Rule (IR), a fact, and two constants (Anna and Bob)
Output: Estimation of the Maximum Probability of Smokes (Anna), Cancer (Bob),
and Cancer (Anna) of being True.
// The Procedure steps
1.Estimate the strength of the IR.
    1.1.  Create all possible worlds based on the given constants (Anna and Bob) and
          the predicate (ground atoms)
    1.2.  Create Truth Table and find the line (Cases) in the Truth Table where the
          target predicate is True
    1.3.  Consider the predicates which have a value equal to True to be a clique.
2. Estimation the maximum probability of the smoking and cancer facts between
friends in a social network based on the given fact which is Bob(B) smokes.
Return: Maximum Probability of each fact.
```

Figure 5.6. Markov Logic Network Procedure.

Figure 5.6 represents the *MLN* Procedure as follows: The input of the *MLN* is a set of The Inference Rules (*IR*) (as in Figure 5.4 and Figure 5.5), a fact [Smokes (*Bob*)], and two constants (*Anna* and *Bob*), where the output is an estimation of the maximum probability of all possible smoking and cancer fact being True.

Figure 5.7 was created where *A* stands for *Anna,* and *B* stands for *Bob*. The predicate that had a True value (which is the truth grounding) became a clique (according to Table 5.8 and Table 5.9).

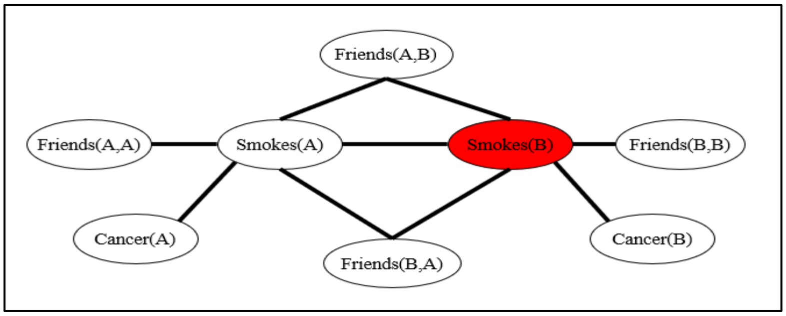


Figure 5.7. The Ground Markov Network for Social Network Example.

The explanation of the *MLN* procedure is as follows:

Step 1: Estimate the Strength of the Inference Rule (*IR*).

In this step, Markov Logic adds strength to each *IR* to indicate the confidence of the knowledge. In other words, the strength reflects how strong the *IR* is. For example, the strength of the generated *IR's* in this example is 1.5 and 1.1, respectively

1.5 Smokes (*x*) => Cancer (*x*)

1.1 Friends (*x, y*) => (smokes (*x*) <=> smokes (*y*)

The mathematical explanation of the used method begins by estimating the strength of the *IR*. In this example, the strengths are estimated by maximizing the pseudo-log-likelihood of the entire set of ground atoms, as the probability distribution over the possible worlds *x* for the Markov Logic Network based on (Richardson & Domingos, 2006) are given by:

$$P_s(x) = \frac{exp(\sum_{i=1}^{n} S_i f_i(x)}{Z} \quad (5.3)$$

where:

$$Z=\sum_{X} S_i f_i\,(x) \tag{5.3.1}$$

Where:

$\boldsymbol{f_i(x)}$: is the number of true groundings (cliques) of the Inference Rule (*IR*), and

$\boldsymbol{S_i}$: is the *IR* strength that needs to be estimated from the given data.

The log-likelihood concerning a particular strength Si is given by Equation 5.3. By taking a log for Equation, 5.3 (Lowd & Domingos, 2007).

$$log\,P_s(X{=}x) = Log\left\{\frac{exp(\sum_{i=1} s_i f_i(x))}{Z}\right\} \tag{5.3.2}$$

$$log\,P_s(X{=}x) = \sum_{i} S_i f_i(x) - log\,Z \tag{5.3.3}$$

A derivative of the log-likelihood of a *IR* with respect to its strength:

$$\frac{\partial}{\partial s_j} log\,P_s(X{=}x) = \frac{\partial}{\partial s_j}\sum_{i} S_i f_i(x) - \frac{\partial}{\partial s_j} log\,Z \tag{5.3.4}$$

$$= f_i(x) - \frac{1}{Z}\frac{\partial}{\partial s_j} Z \tag{5.3.5}$$

$$= f_i(x) - \frac{1}{Z}\sum x' \frac{\partial}{\partial s_j} exp\left(\sum_{i} S_i f_i(x')\right) \tag{5.3.6}$$

$$= f_i(x) - \frac{1}{Z}\sum x' \frac{\partial}{\partial s_j} exp\left(\sum_{i} S_i f_i(x')\right) s_i(x') \tag{5.3.7}$$

Derivative of $s_i f_i(x)$ with respect to S_i is zero for i != j

$$f_i(x) - \frac{1}{Z}\sum x' \frac{\partial}{\partial s_i} exp\left(\sum_{i} S_i f_i x'\right) f_i(x') = 0 \tag{5.3.8}$$

Then the update *IR* with gradient ascent for S_i is:

$$f_i(x) - \frac{1}{Z}\sum x' \frac{\partial}{\partial s_i} exp\left(\sum_i S_i f_i x'\right) f_i(x') = 0 \qquad (5.3.9)$$

Applying the steps above for this example, the goal is to estimate of the strength of the *IR's* in Figure 5.4 and Figure 5.5 by using Equation 5.3.2.

S Smokes (*x*) => Cancer (*x*)
1.5 Smokes (*x*) => Cancer (*x*)

S Friends (*x, y*) => (smokes (*x*) <=> smokes (*y*))
1.1 Friends (*x, y*) => (smokes (*x*) <=> smokes (*y*)

As shown above, the strength of the *IR* is 1.5 and 1.1 respectively. The steps can then be shown to estimate the strength of each *IR*.

Step 1.1: Create all outcomes (atomic formulas) based on the given constants and facts. Consider that constants (*Anna* and *Bob*), one fact [smokes (*Bob*)] and two *IR's* [Friends (*x*, *y*) => (smokes (*x*) <=> smokes (*y*)] and [Smokes (*x*) => Cancer (*x*)] are given to create all possible outcomes as in Table 5.3. There are 2 entities (Constants): (*Anna* and *Bob*).

The *IR* has three facts (Friends, smokes, Cancer), all of which are binary relations from propositional logic(Cheng, Wan, Buckles, & Huang, 2014; Mario&Matr, 2014; Urbanek & Theus, 2008). The number of possible ground atoms for each relation = n^r., where *n*: is a number of constants, and *r*: is the number of relations. Then, the number of possible ground atoms for each relation = 2^3 = 8.

Table 5.3 All Possible Ground Atoms.

Friends
Friends (*Ana*, *Ana*)
Friends (*Ana*, *Bob*)
Friends (*Bob*, *Bob*)
Friends (*Bob*, *Ana*)
Cancer
Cancer (*Ana*)
Cancer (*Bob*)
Smokes
Smokes (*Ana*)
Smokes (*Bob*)

Step 1.2: Create a Truth Table for each atomic formula and find the line (Cases) in the Truth Table where the target predicate is True.

In Step 1.1, all possible atomic formulas were generated; in this step, the truth table was created for each atomic formula. The truth table for all atomic formulas are in Table 5.4, Table 5.5, Table 5.6, and Table 5.7

Table 5.4. The Truth Table for Atomic Formula 1

Smokes (*Anna*)	Cancer (*Anna*)	*IR* (**If-Then**)	Potential Function *f(x)*
0	0	1	1
0	1	1	1
1	0	0	0
1	1	1	1

Table 5.5. The Truth Table for Atomic Formula 2

Smokes (*Bob*)	Cancer (*Bob*)	*IR* (**If-Then**)	Potential Function *f(x)*
0	0	1	1
0	1	1	1
1	0	0	0
1	1	1	1

Table 5.6. The Truth Table for Atomic Formula 3

Friends (*Anna, Bob*)	Smokes (*Anna*)	Smokes (*Bob*)	*IR*		Potential Function *f(x)*
1	1	0	0	0	0
1	1	1	1	1	1
1	0	0	1	1	1
1	0	1	0	0	0
0	1	0	0	1	1
0	1	1	1	1	1
0	0	0	1	1	1
0	0	1	0	1	1

Table 5.7. The Truth Table for Atomic Formula 4

Friends (*Bob, Anna*)	Smokes (*Anna*)	Smokes (*Bob*)	*IR*		Potential Function *f(x)*
1	1	0	0	0	0
1	1	1	1	1	1
1	0	0	1	1	1
1	0	1	0	0	0
0	1	0	0	1	1
0	1	1	1	1	1
0	0	0	1	1	1
0	0	1	0	1	1

Finding the line (Cases) in the Truth Table where the target predicate is True for the IR's in Figure 5.4 and Figure 5.5 are illustrated in Table 5.8 and Table 5.9.

Table 5.8. The potential Function for Atomic Formula 1 and 2.

Potential Function Atomic Formula 1	Potential Function Atomic Formula 2	$f_i(x)$	$e^{f_i(x)*s_i}$
1	1	2	e^{2s}
1	1	2	e^{2s}
0	0	0	1
1	1	2	e^{2s}

Table 5.9. The potential Function for Atomic Formula 3 and 4.

Potential Function Atomic Formula 3	Potential Function Atomic Formula 4	$f_i(x)$	$e^{f_i(x)*s_i}$
0	0	0	1
1	1	2	e^{2s}
1	1	2	e^{2s}
0	0	0	1
1	1	2	e^{2s}
1	1	2	e^{2s}
1	1	2	e^{2s}
1	1	2	e^{2s}

Let us start calculating the strength for the *IR* in Figure 5.4 as follows:

$$P_s(x) = \frac{exp(\sum_{i=1}^{n} S_i f_i(X)}{Z}$$

where*:*

$$Z = \sum_X S_i f_i\,(x)$$

First, let us calculate the **Numerator** in the above Equation

$$P_s(x) = \frac{exp(2s+2s+1+2s)}{Z} \qquad (5.3.10)$$

$$P_s(x) = \frac{(e^{2s}.e^{2s}.1.e^{2s})}{Z} \qquad (5.3.11)$$

$$P_s(x) = \frac{(e^{2s}.e^{2s}.1.e^{2s})}{Z} = \frac{(e^{2s})^3}{Z} = \frac{(e^{6s})}{Z} \qquad (5.3.12)$$

Second let us calculate the **Denominator** in Equation 5.3.

Since $Z = \sum_X S_i f_i\,(x)$

Then*:*

$$Z = 3\,e^{2s} + 1 \qquad (5.3.13)$$

By substituting in Equation 5.3:

$$P_s(x) = \frac{e^{6s}}{3\,e^{2s} + 1} \qquad (5.3.14)$$

As seen in Equation 5.3.14 the Equation cannot be computed in closed form, but it can be found using an optimization gradient descent method known as the Broyden-Fletcher-Goldfarb-Shanno (*BFGS*) optimization algorithm (Biegler, 2010; Hasdorff, 1976; Jahn, 2007). A Python scikit package was also used to calculate it (Pedregosa et al., 2011). The ground formulae created from the same *IR* shared their strengths.

Quasi-Newton methods are well-known methods in solving optimization problems.

These methods, which use the updating formulas for approximation of the Hessian, were introduced by Davidon in 1959, and later popularized by Fletcher and Powell in 1963 to give the Davidon-Fletcher-Powel (*DFP*) method. In 1970 Broyden, Fletcher, Goldfarb and Shanno developed the idea of a new updating formula, known as *BFGS*, which has become widely used in many researches. In this dissertation the *BFGS* method was used to estimate the strength of the *IR*, which is equal to 1.5. By applying the same steps for the *SIR* in Figure 5.5.

1.5 Smokes (*x*) => Cancer (*x*)

As seen in Equation 5.3.19 the Equation cannot be computed in closed form, but by using the *BFGS* method the strength of the *IR* is equal to 1.1

$$P_s(x) = \frac{exp(\sum_{i=1}^{n} S_i f_i(X)}{Z}$$

where:

$$Z = \sum_{X} S_i f_i\,(x)$$

First, let us calculate the **numerator** in the above Equation

$$P_s(x) = \frac{exp(1+2s+2s+1+2s+2s+2s+2s)}{Z} \qquad (5.3.15)$$

$$P_s(x) = \frac{(1.e^{2s}.e^{2s}.1.e^{2s}.e^{2s}.e^{2s}.e^{2s})}{Z} \qquad (5.3.16)$$

$$P_s(x) = \frac{(1.e^{2s}.e^{2s}.1.e^{2s}.e^{2s}.e^{2s}.e^{2s})}{Z} = \frac{(e^{2s})^{6}}{Z} = \frac{(e^{12s})}{Z} \qquad (5.3.17)$$

Second let us calculate the **denominator** in Equation 5.3.

Since $Z = \sum_X S_i f_i\,(x)$ then:

$$Z = 3\ e^{2s} + 1 \tag{5.3.18}$$

By substituting in Equation 5.3

$$P_s(x) = \frac{e^{12s}}{6\ e^{2s}\ +\ 2} \tag{5.3.19}$$

1.1 Friends (*x, y*) => (smokes (*x*) <=> smokes (*y*)

Step 1.3: Consider the predicates which require a value equal to be True in order to be a clique.

In the second step, the possible world for each atomic formula was created by using truth tables. The result of that process is called an observation. After the ground predicates are identified, it should be transformed to a graph where nodes are the facts, and edges between them are added if two ground facts appear in the same *IR*. In this case, the groundings of the predicates were {Smokes (*Bob*), Smokes (*Anna*), Cancer (*Bob*), Cancer (Anna), friends (*Bob*, *Anna*), friends (*Anna*, *Bob*), friends (*Bob*, *Bob*), friends (Anna, Anna)}. According to the atomic formulas in Table 5.3, the ground Markov Network in

Step 2: Estimate the maximum probability of the smoking and cancer facts between friends in a social network based on the given fact which is *Bob*(*B*) smokes.

What the estimate means in this context was the act of querying the Markov Logic Network. Since the *MLN* could represent the full joint probability distribution over the set of random variables, it could then be used to answer any probabilistic query about the world (de Oliveira, 2009; Getoor & Taskar, 2007). There are two common types of queries used for different purposes:

- **The Conditional Probability Query** in the form *p(Y|E=e)*, where *Y* is the query variable and *E* is the evidence. That is, the posterior probability distribution over the values *y* of *Y* is conditioned on the fact that *E = e*. For example, *P* (Beach | Sunny=true, Hot=true) gives two probabilities distributions: one when Beach=true and another when Beach=false;
- The **Maximum a Posteriori Query (*MAP*)** is a query in the form $argmax_y(y|e)$. This type is used when calculating not only the probability of a set of variables *Y*, given the evidence *E=e* is needed, but for which that probability is maximal as well. **For example**, a *MAP* query of Beach, given {Sunny=true, Hot=true} provides the most likely assignment to the variable Beach (de Oliveira, 2009).

In the inference of the *MLN*, the Maximum A Posteriori (*MAP*) is used. This type of reasoning is called an approximate inference. Different types of algorithms are utilized to perform this type of reasoning (Koller & Friedman), one of the simplest of which is a Markov Chain Monte Carlo (*MCMC*) (Metropolis, Rosenbluth, Rosenbluth, Teller, & Teller, 1953).

A Markov Chain Monte Carlo (*MCMC*) algorithm is used to obtain a sequence of observations which are approximated from a specified probability distribution when direct sampling is difficult. This sequence can be used to approximate the conditional probability.

One of the algorithms *MCMC* uses for sampling is the Gibbs Sampling Algorithm. The Gibbs Sampler was first formally introduced by (Casella & George, 1992; Geman & Geman, 1984; Robert, 2014) to the field of image processing. The Gibbs Sampling Algorithm is simple: it continuously samples a variable from its posterior distribution with all other variables temporally fixed. After a long while, the samples are guaranteed to be unbiased. To accomplish this goal, Markov chains are used as a fundamental method (Hastings, 1970). Generating a Markov chain makes sense of using the previous sample values to randomly generate the next sample value. It can be seen as a transition probability between sample values. The steps of the Gibbs Sampling procedure are explained in Figure 5.8. There are two essential components of a Markov chain: the states, and the transition probability.

Procedure 2 Gibbs Sampling

1. Start with an initial random assignment to nodes
2. One node at a time, sample node given its Markov blanket
3. Repeat
4. Use samples to compute *P(X)*
5. Apply to the ground network

Figure 5.8.Gibbs Sampling Procedure.

The **Gibbs** Sampling Algorithm can begin to be applied to the Markov Logic Network (as in Figure 5.7) since it is a simple graph that is used to estimate the maximum probability of each fact of being True. By using the inference task, the proximate result for our query of the smoking habit and having cancer in our small social network which is Anna and Bob will be as follows: The probability that Anna will have cancer is 51%, and the probability that Bob will have cancer is 52%, and Anna will be a smoker is 52% with our evidence that Bob smokes.

5.5 A Markov Cognitive Knowledge State Network (*MCKSN*)

Definition: A Markov Cognitive Knowledge State Network (*MCKSN*): is defined as an undirected graph *G= (F, R)*, where each node F_i in the node set *F* represents a Cognitive Skill Dependency at a given Bloom level. $R \subseteq F \times F$ is a set of edges connecting the nodes. Each edge $r_{i,j}$ represents the appearance of Cognitive Skill Dependency in the same Skill Inference Rule(*SIR*).

The nodes in an MSKSN represent the Cognitive Skill Dependency, where the Cognitive Skill Dependency are equivalent to the random variables in *MN*. In most domains, it is challenging to come up with a Cognitive Skill Dependency that is always true that is the reason for modeling random variables to Cognitive Skill Dependencies. An edge connects two Cognitive Skill Dependencies if both of them appear in the same Skill Inference Rule (*SIR*) where the edge represents the dependency relationship between them. If the *SIR* combines the connection between a few Cognitive Skill Dependencies, one could influence the other dependently.

The Cognitive Skill Dependencies in an *MSKS* should have the same assumptions as those in an *MN*. More than one Cognitive Skill Dependency can represent an *SIR*. The *SIR's* are expressed according to the syntax of the logical language. The primary goal of *SIR* is to infer new Cognitive Skill Dependencies from existing ones. One example is fatherOf (x, y) >> sonOf (y, x); in every *SIR*, if x is the father of y, it is implied that y is also the son of *x*. In this example, the inference is defined as a technique to infer a new fact from the *SIR*. If it is known that *x* is the father of *y*, then y also being the son of *x* could be easily inferred.

Input:

1. **Constants** CS concept space where the concepts constructed from Textbooks.
2. **Cognitive Skill Dependencies**: is a logical fact generated by mapping the concepts from a set of terms in a sentence to the set of nodes using verb-based mapping.
3. **Skill Inference Rules (*SIR*):** is defines as a logical relationship between a set of Cognitive Skill Dependencies (*CSDs*) $d_i=e\ (a_i,\ b_i,\ BL_i)$. The logical relationship between any set of *CSDs* can be expressed as a first order logic expression. More formally an example of that $\phi_i = \forall_{A,B,C} \{e(A, B, BL_i) \wedge e(B, C, BL_i) => e(C, A, BL_i)\}$. In other words, if concept *A* is needed to learn concept *B,* and concept *C* is needed to learn *B*, then concept *C* is needed to learn concept *A*.
4. The **potential function** in this example is composed of simple truth tables for the *SIR*, expressed with integers {0, 1}. For each *SIR*, this function assumes a value of 1 for a state of the clique if the truth values of the nodes make the *SIR* true, while considering a value of 0 for a state of the clique if the truth values of the nodes make the *SIR* false.
5. **User Query:** Find out the remaining Cognitive Skill Dependencies (*CSD*) between concepts.

Output:

MSKSN can estimate the maximum probability of the Cognitive Skill Dependencies (*CSD*) between concepts.

5.5.1 Example of Markov Cognitive Knowledge State Network (*MCKSN*)

The motivating example is used to explain the procedure of applying *MLN*. Suppose the *SIR* in Figure 5.9 is one of the *ISR's* generated in Chapter Four. The *SIR* means that IF the concept *z* is essential to be learned to Create Concept *x, AND* if it is also important to be learned to Apply concept *y*, then concept *x* is important to be learned to Create *y*, where two different cognitive Skill Dependencies*(CSD)* are given. Consider that three CS concepts (*x*, *y*, and *z*) are given (the number of concepts is limited to three to simplify the calculation so that it can be easy to follow) and consider that three concepts from CS knowledge space are used: Graph, Graph-Traverse, and Depth First Search (*BFS*). Also suppose that a *SIR* extracted from *SKM* (as mentioned in Chapter Four), as well as one *CSD* as evidence is given (as in Figure 5.10). The '*Graph'* as a concept would then be important to be learned in order to apply 'Graph-Traversal'.

Create (*z, x*) ^ Apply (*z, y*) => Create (*x, y*)

Figure 5.9. Skill Inference Rule (SIR) Template.

Apply ('*Graph'*, '*BFS'*)

Figure 5.10. The Cognitive Skill Dependency (Apply)

The question is how to estimate the probability of knowing one of the given concepts to Apply and Create the other concept based on the given *CSD* that to apply the

Breadth-first Search (*BFS*) concept, one must know the Graph concept as well. The proposed solution is to use a Markov Cognitive Knowledge State Network (*MCKSN*), Model. The main steps of the proposed *MLN* model are illustrated in Figure 5.11.

```
Procedure 3:  Markov Cognitive Knowledge State Network (MCKSN)
Input: subset of CSD's, a set of SIR, and three CS-concepts ('Graph,' 'Graph-Traverse,'
       and 'Depth First Search')
Output: Estimation of the Maximum Probability of each CSD of being True
// The Procedure steps
    1.   Estimate the strength of the SIR.
    1.1. Create all possible worlds based on the given constants (CS-concepts) and the
         predicate (ground atoms)
    1.2. Create Truth Table and find the line (Cases) in the Truth Table where the target
         predicate is True
    1.3. Consider the predicates which have a value equal to True to be a clique.
2.  Estimate the maximum probability of the knowing state of a given CSD at a
    particular bloom level given the states of few other concepts at a specific Bloom
    level (SIR).
Return: Maximum Probability of each Bloom Fact.
```

Figure 5.11. MCKSN Procedure.

The algorithm in Figure 5.11 represents the *MCKSN* procedure as follows: The input of the algorithm is a *SIR* (as in Figure 5.9), a *CSD* (as in Figure 5.10), and three CS-concepts *(Graph, Graph-Traverse, and Depth First Search),* where the output is an estimation of the maximum Probability of all possible *CSD's* being True. The explanation of the *MCKSN* procedure is as follows:

Step 1: Estimate the strength of the *SIR*.

In this step, *MCKSN* adds strength to each *SIR* to indicate the confidence of the knowledge. In other words, the strength reflects how strong the *SIR* is. For example, the strength of the given *SIR* in the following example is 12.69.

12.69 Create (x, y) ^ Apply (y, z) => Create (x, z)

The mathematical explanation of the used method begins by estimating the strength of the *SIR*. In this dissertation, the strengths are evaluated by maximizing the pseudo-log-likelihood of the entire set of ground atoms, as the probability distribution over the possible worlds x for the *MCKSN* are given by:

$$P_s(x) = \frac{exp(\sum_{i=1}^{n} S_i f_i(X))}{Z} \qquad (5.3)$$

Where:

$$Z = \sum_{X} S_i f_i(x) \qquad (5.3.1)$$

Where:

$\boldsymbol{f_i(x)}$: is the number of true groundings (cliques) of the *SIR*, and

$\boldsymbol{S_i}$: is the strength of *SIR* that needs to be estimated.

The log-likelihood with respect to a particular strength S_i is given by Equation 5.3. By taking a log for Equation, 5.3. Assuming the following *SIR* as an example, the goal is to estimate the strength of the *SIR* in Figure 5.9 by using Equation 5.3.

S (CREATE (x, y) ^ APPLY (y, z)) => CREATE (x, z)

12.69 (CREATE (x, y) ^ APPLY (y, z)) => CREATE (x, z)

As shown above, the strength of the *SIR* is 12.69. The steps can then be shown to estimate the strength of the *SIR*.

Step 1.1: Create all possible outcomes (atomic formulas) based on the given constants and *CSD*. Consider that three concepts *(Graph, Graph-Traversal, and BFS),* one *CSD* [Apply (Graph, BFS)] and a *SIR* [*Create (z, x)* ^ *Apply (z, y)* => *Create (x, y)*] are given to create all possible outcomes. There are three entities (Constants): Graph, Graph Traversal, and BFS.

- The *SIR* has two relations (*Create*, *Apply*), both of which are binary relations from propositional logic.
- The number of possible ground atoms for each relation = n^r., where *n*: is a number of constants, and *r*: is the number of relations.
- Then, the number of possible ground atoms for each relation = 3^2 = 9.
- The total number of ground atoms = 9✕3 =27 ground atoms.

Table 5.10 All Possible Ground Atoms for the Example.

Créate (X,Y) ^Apply (Y,Z) => Create (X,Z)			
Z	X	Y	
Graph	Graph	Graph	1. CR (Graph, Graph) ^AP(Graph, Graph)) => CR (Graph, Graph)
	Graph	BFS	2. CR (Graph, Graph) ^AP (Graph, BFS) => CR (Graph, BFS)
	Graph	Graph Traverse	3. CR (Graph, Graph) ^AP (Graph, GT) => CR (Graph, GT)
	BFS	BFS	4. CR (Graph, BFS) ^AP(Graph, BFS) => CR (BFS, BFS)
	BFS	Graph	5. CR (Graph, BFS) ^AP(Graph, GT) => CR (BFS, GT)
	BFS	Graph Traverse	6. CR(Graph, Graph)^AP(Graph, Graph)) => CR (Graph, Graph)
	GT	Graph Traverse	7. CR(Graph, GT)^AP(Graph, GT)) => CR (GT,GT)
	GT	BFS	8. CR(Graph, GT)^AP(Graph, BFS)) => CR (GT, BFS)
	GT	Graph	9. CR(Graph, GT)^AP(Graph, Graph)) => CR (GT, Graph)
BFS	BFS	BFS	10. CR(BFS, BFS)^AP(BFS, BFS)) => CR (BFS, BFS)
	BFS	Graph	11. CR(BFS, BFS)^AP(BFS, Graph))) => CR (BFS, Graph)
	BFS	Graph Traverse	12. CR(BFS, BFS)^AP(BFS, GT) => CR (BFS, GT)
	Graph	BFS	13. CR(BFS, Graph)^AP(BFS, BFS)) => CR (Graph, BFS)
	Graph	Graph	14. CR(BFS, Graph)^AP(BFS, Graph)) => CR (Graph, Graph)
	Graph	Graph Traverse	15. CR(BFS, Graph)^AP(BFS, GT)) => CR (Graph,GT)
	GT	Graph Traverse	16. CR(BFS, GT)^AP(BFS, GT)) => CR (GT,GT)
	GT	BFS	17. CR(BFS, GT)^AP(BFS, BFS)) => CR (GT, BFS)
	GT	Graph	18. CR(BFS, GT)^AP(BFS, Graph)) => CR (GT,Graph)
Graph Traverse	GT	Graph Traverse	19. CR (GT, GT)^AP(GT, GT) => CR (GT, GT)
	GT	BFS	20. CR(GT, GT)^AP(GT, BFS) => CR (GT, BFS)
	GT	Graph	21. CR (GT, GT)^AP(GT, Graph) => CR (Graph, GT)
	BFS	BFS	22. CR(GT, BFS)^AP(GT, BFS) => CR (BFS, BFS)
	BFS	Graph	23. CR (GT, BFS) ^AP (GT, Graph) => CR (BFS, Graph)
	BFS	Graph Traverse	24. CR (GT, BFS)^AP(GT, GT) => CR (BFS,GT)
	Graph	BFS	25. CR (GT, Graph) ^AP (GT, BFS)) => CR (Graph, BFS)
	Graph	Graph	26. CR(GT, Graph)^AP(GT,Graph)) => CR (Graph, Graph)
	Graph	Graph Traverse	27. CR (GT, Graph) ^AP (GT, GT) => CR (Graph, GT)

For formatting purposes, the concepts and the relations are abbreviated to letters such as: Graph-Traversal = GT, Breadth-First Search = BFS, create =CR, and Apply= AP.

Step 1.2: Create a Truth Table and find the line (Cases) in the Truth Table where the target predicate is True.

In Step 1.1, all possible atomic formulas were generated; in this step, the truth table was created for each atomic formula. Truth tables are constructed from *SIR's* by transforming expressions into atomic formulas. To follow the next steps more easily, only three atomic formulas were used. These three atomic formulas were three rows in the third column, each of which is highlighted by different colors (as in Table 5.10). Table 5.11 shows the chosen atomic formulas. Tables 5.12, Table 5.13, and Table 5.14 illustrate the truth tables for the three atomic formulas with their respective coded colors.

Table 5.11. The Chosen Atomic Formulas.

Atomic Formulas
1. CR (Graph, BFS) ^AP (Graph, GT) => CR (BFS, GT)
2. CR (Graph, GT) ^AP (Graph, BFS) => CR (GT, BFS)
3. CR (BFS, GT) ^AP (Graph, BFS) => CR (GT, Graph)

Table 5.12. The Truth Table for Atomic Formula 1.

CR (Graph, BFS)	AP (Graph, GT)	CR (BFS, GT)	CR (Graph, BFS) ^AP (Graph, GT)	*SIR* (**If-Then**)	Potential Function *f(x)*
1	1	1	1	1	1
1	1	0	1	0	0
1	0	1	0	1	1
1	0	0	0	0	0
0	1	1	0	1	1
0	1	0	0	0	0
0	0	1	0	1	1
0	0	0	0	0	0

Table 5.13. The Truth Table for Atomic Formula 2.

CR (Graph, GT)	AP (Graph, BFS)	CR (GT, BFS)	CR (Graph, GT) ^AP (Graph, BFS)	*SIR* **(If-Then)**	Potential Function *f(x)*
1	1	1	1	1	1
1	1	0	1	0	0
1	0	1	0	1	1
1	0	0	0	0	0
0	1	1	0	1	1
0	1	0	0	0	0
0	0	1	0	1	1
0	0	0	0	0	0

Table 5.14. The Truth Table for Atomic Formula 3.

CR (BFS, Graph)	AP (Graph, BFS)	CR (GT, Graph)	CR (BFS, GT) ^AP (Graph, BFS)	*SIR* **(If-Then)**	Potential Function *f(x)*
1	1	1	1	1	1
1	1	0	1	0	0
1	0	1	0	1	1
1	0	0	0	0	0
0	1	1	0	1	1
0	1	0	0	0	0
0	0	1	0	1	1
0	0	0	0	0	0

Where the potential function is defined as:

$$f(x) = \begin{cases} 1 \text{ if } \quad SIR \text{ is true(Has value=1 in the truth table)} \\ 0 \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \text{otherwise} \end{cases}$$

Finding the line (Cases) in the Truth Table where the target predicate is True for the *SIR* in Table 5.12, Table 5.13 and Table 5.14 illustrated in Table 5.15. This is achieved by applying the Equation 5.3 for the three atomics formulas used (as in Table 5.15).

Table 5.15. The Truth Table for Atomic Formula 1,2, and 3.

Potential Function Atomic Formula 1	Potential Function Atomic Formula 2	Potential Function Atomic Formula 3	$f_i(x)$	$e^{f_i(x)*s_i}$
1	1	1	3	e^{3s}
0	0	0	0	1
1	1	1	3	e^{3s}
0	0	0	0	1
1	1	1	3	e^{3s}
0	0	0	0	1
1	1	1	3	e^{3s}
0	0	0	0	1

It is started by calculating the strength for the *SIR* in Figure 5.9 as follows:

$$P_s(x)=\frac{exp(\sum_{i=1}^{n} S_i f_i(X)}{Z}$$

where:

$$Z=\sum_{X} S_i f_i\,(x)$$

First, calculate the **Numerator** in the above Equation

$$P_s(x)=\frac{exp(3s+1+3s+1+3s+1+3s+1)}{Z} \tag{5.3.19}$$

$$P_s(x)=\frac{(e^{3s}.1.e^{3s}.1.e^{3s}.1.e^{3s}.1)}{Z} \tag{5.3.20}$$

$$P_s(x)=\frac{(e^{3s}.1.e^{3s}.1.e^{3s}.1.e^{3s}.1)}{Z}=(e^{3s})^4 = e^{12s} \tag{5.3.21}$$

Second calculate the **Denominator** in Equation 5.3.

Since $Z=\sum_X S_i f_i\,(x)$

Then:

$$Z = 4\,e^{3s} + 4 \tag{5.3.22}$$

By substituting in Equation 5.3

$$P_s(x) = \frac{e^{12s}}{4\;e^{3s} + 4} \tag{5.3.23}$$

As seen in Equation 5.3.23 the Equation cannot be computed in closed form, but using *BFGS* method the strength of the *SIR* is equal to 12.69

12.69 (Create (*z*, *x*) ^ Apply (*z*, *y*)) => Create (*x*, *y*)

Step 1.3: Consider the predicates which require a value equal to True to be a clique. In the second step, the possible outcomes for each atomic formula were created by using truth tables. After the ground predicates are identified, it should be transformed to a graph where nodes are the ground predicates, and edges between them are added if two ground atoms appear in the same *SIR*. In this case, the groundings of the predicates were {*CR (Graph, BFS), AP (Graph, GT), CR (BFS, GT), CR (Graph, GT), AP (Graph, BFS), CR (GT, BFS), CR (GT, Graph)}*. The ground Markov Network in Figure 5.12 was created. The predicate that had a True value (which is the truth grounding) became a clique.

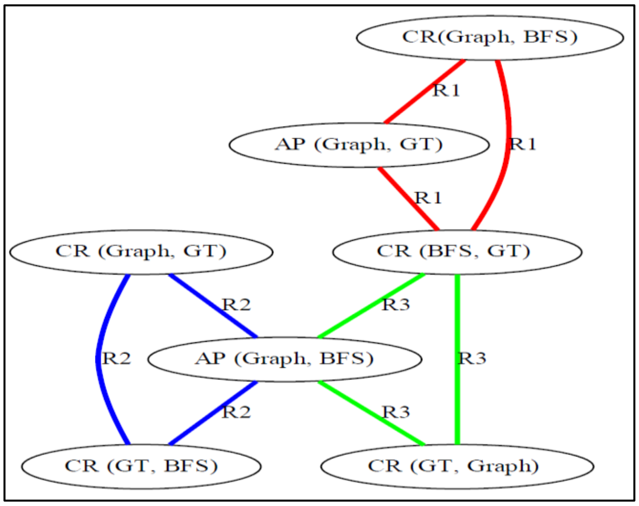


Figure 5.12. The Ground Markov Cognitive Knowledge State Network (MCKSN).

Step Two: Estimate the maximum probability of the known state of a given concept at a particular Bloom level, given the states of a few other concepts at a particular Bloom level. In the inference of the *MCKSN*, the Maximum A Posteriori (*MAP*) is used. This type of reasoning is called an approximate inference. Different types of algorithms are utilized to perform this type of inference (Koller & Friedman), one of the simplest of which is a Markov Chain Monte Carlo (*MCMC*) (Stuart Russell and Norvig 2002).

A Markov Chain Monte Carlo (*MCMC*) algorithm is introduced in section 5.3. The steps of the Gibbs Sampling Algorithm are explained in Figure 5.8. The Gibbs Sampling Algorithm can begin to be applied to the *MCKSN* (as in Figure 5.12) since it is a Sub-network that is used to estimate the maximum probability of each cognitive fact of being True.

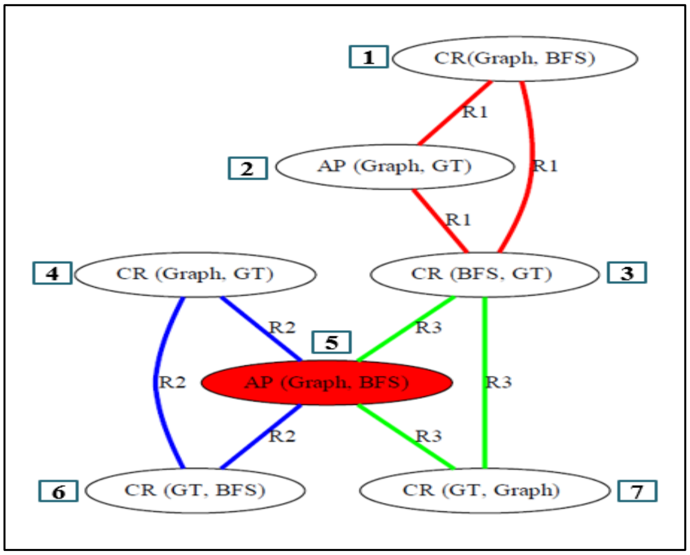


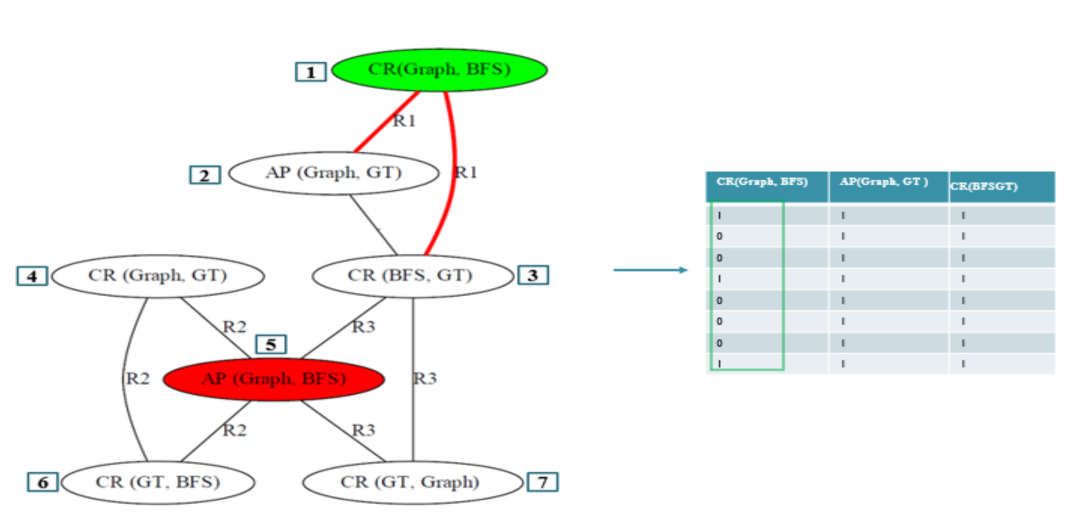
Figure 5.13. A Sub-Network MLN.

The estimation of the Maximum Probability is done iteratively, and each *CSD* in the graph is also updated iteratively (according to the probability of each *CSD*, given its Markov blanket). When the first iteration is finished over all the nodes, then one cycle is completed. In this example, the starting node was node Number 1, the ground atom CR (Graph, BFS). The ground atom holds the cognitive relationship Create (CR) between two concepts Graph and BFS. So, estimating the maximum probability value of this Bloom fact to be true needs to be computed. The Red node AP (Graph, BFS) is given as a fact in this example. Considering that each node corresponds to a random variable, the probability with which each node is sampled can be calculated based only on its Markov blanket. The likelihood of any ground atom x when its Markov blanket *MB(x)* is estimated can be shown by using Equation 5.4:

$$P(x|MB(x))=\frac{exp(\sum_i s_i f_i(x))}{exp(\sum_i s_i f_i(x=0))+exp(\sum_i s_i f_i(x=1))} \tag{5.4}$$

Where f_i is the set of ground formulas that x appears in, the *MB(x)* Markov blanket of node *x*, and $f_i(x=1)$ and $f_i(x=0)$ are the values (0 or 1) of the feature corresponding to the ith ground formula.

Figure 5.14 requires estimating the probability value for the green node. Based on Equation 5.4, the maximum probability could be used for the conditional green node in its Markov blanket, the observations of which are in the attached truth table in Figure 5.14. Since the green node was node number 1, it had two neighbors in the same clique. Sampling node number 1 from its posterior distribution with all other variables temporally fixed with the same value was done by applying Equation 5.4.



CR(Graph, BFS)	AP(Graph, GT)	CR(BFSGT)
1	1	1
0	1	1
0	1	1
1	1	1
0	1	1
0	1	1
0	1	1
1	1	1

Figure 5.14. Gibbs Sampling for Node Number 1.

A calculation of *P (CR (Graph, BFS)* using Equation 5.4 goes as follows:

$$P(x|MB(x))=\frac{exp(\sum_i s_i f_i(x))}{exp(\sum_i s_i f_i(x=0))+exp(\sum_i s_i f_i(x=1))}$$

First, calculate the **Numerator** in Equation 5.4

$exp(\sum_i s_i f_i(x))$ counts all the assignments where CR (Graph, BFS) is True.

This is based on the generated sampling using Gibbs sampling. Sample *P (CR (Graph, BFS))* given its Markov blanket, and repeat. Count the number of times that *P (CR (Graph, BFS))* is true and false in the samples. As illustrated in Figure 5.14, the sampling is generated randomly, where

AP (Graph, GT) is equal to:

$$exp\left(\sum_i s_i f_i(x)\right) = (3*e^{12S}) = 3e^{12S}$$

The **denominator** in Equation 5.4, is $exp\left(\sum_i s_i f_i(x=0)\right) + exp(\sum_i s_i f_i(x=1))$

In Equation 5.4 $exp\left(\sum_i s_i f_i(x=0)\right)$ represents the cases where CR (Graph, BFS) is False conditional of its *MB* (their first level neighbor), which is also is False. That means

$$P\ (CR\ (Graph,\ BFS)\ =0\ |\ AP\ (Graph,\ GT)\ =0,\ CR\ (BFS,\ GT)\ =0) = 5.$$

In Equation 5.4 $exp\left(\sum_i s_i f_i(x=1)\right)$ represents the cases where *CR (Graph, BFS)* is a True conditional of its *MB*, which is also a True. That means

$P\ (CR\ (Graph,\ BFS)\ =1\ |\ AP\ (Graph,\ GT)\ =1,\ CR\ (BFS,\ GT)\ =1) = 3{*}e^{12S} = 3e^{12S}$

Then, by substituting in Equation 5.4:

$$P\ (CR\ (Graph,\ BFS)\ = \frac{e^{12S}}{5+\ 3\ e^{3s}} \qquad (5.3.11)$$

It is clear that this Equation had multiple solutions, meaning that the Equation was not in a closed form. Therefore, iterative techniques to compute the maximum values should have been used. One of the most straightforward iterative methods is the gradient ascent (as introduced in Section 5.3.1.3):

$$\frac{3e^{12}}{5+3e^{12}} = 0.235004$$

Then the probability of the fact's validity that (node number1) is equal to 0.235004 (as mentioned in the Gibbs sampling) was not the final value. In this case, the sampling was repeated iteratively until the best value was reached.

The next node is node number 2, which is AP (Graph, GT) (as in Figure 5.15). The same steps were repeated to estimate the maximum degree of credibility that learning a concept graph is necessary to know in order to reach the applying cognitive level where the GT concept can be applied. Figure 5.10 explains the method for computing the estimation probability for the green node. As seen in Figure 5.10, the node only had two neighbors in the same clique, so the sampling started by sampling this node where all other variables were temporally fixed with the same value. A calculation of *P (AP (Graph, GT))* using Equation 5.4 goes as follows:

$$P(x|MB(x))=\frac{exp(\sum_i s_i f_i(x))}{exp(\sum_i s_i f_i(x=0))+exp(\sum_i s_i f_i(x=1))}$$

First, the numerator in Equation 5.4 is calculated as $exp\left(\sum_i S_i f_i(x)\right)$, which counts all the assignments where *P (AP (Graph, GT))* is True.

This is based on the generated sampling using Gibbs sampling. Sample *P (AP (Graph, GT))* is given its Markov blanket, and the process is repeated for each assignment. Count the number of times that *P (AP (Graph, GT))* is true and false in the samples. As illustrated in Figure 5.9, the sampling is generated randomly, where *AP (Graph, GT)* is equal to:

$$exp\left(\sum_i S_i f_i(x)\right) = (5*e^{12S}) = 5e^{12S}$$

The **denominator** in Equation 5.4. is $exp\left(\sum_i s_i f_i(x=0)\right) + exp(\sum_i s_i f_i(x=1))$

In Equation 5.4, $exp(\sum_i s_i f_i(x=0))$ represents the cases where P (AP (Graph, GT)) is False, conditional of its *MB* (their first level neighbor) being False too. That means

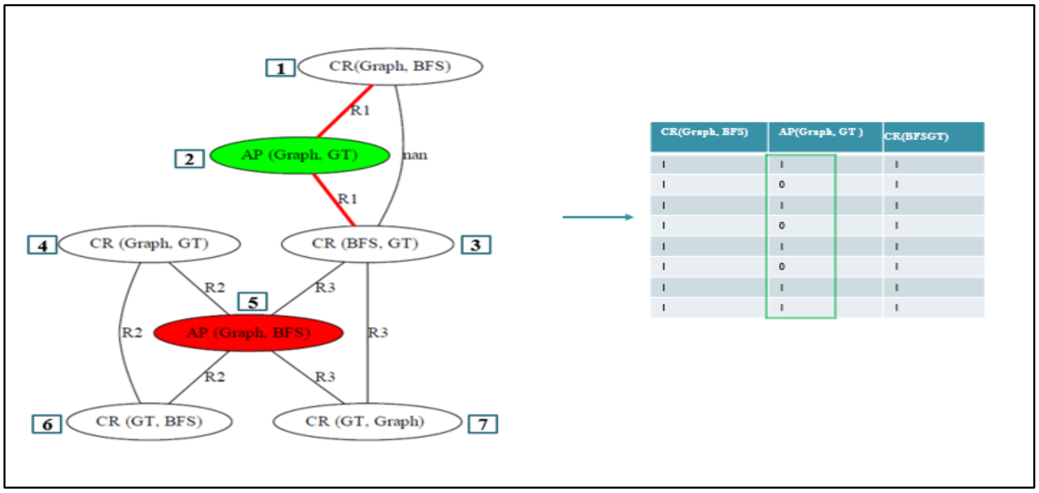
$$P\ (AP\ (Graph,\ GT) = 0|CR\ (BFS,\ GT) = 0,\ CR\ (Graph,\ BFS) = 3.$$

In Equation 5.4, $exp\left(\sum_i s_i f_i(x=1)\right)$ represents the cases where *P (AP (Graph, GT))* is True, conditional of its MB being True too. That means

$$P\ (AP\ (Graph,\ GT) = 1\ |CR\ (BFS,\ GT) = 1,\ CR\ (Graph,\ BFS) = 1 = 5*e^{12S} = 5e^{12S}$$

Then, by substituting in Equation 5.4, the maximum probability of the fact of learning Graph to apply GT is the following:

$$P\ (AP\ (Graph,\ GT) = \frac{5e^{12S}}{3+\ 5\ e^{12s}} \qquad (5.3.11)$$



CR(Graph, BFS)	AP(Graph, GT)	CR(BFSGT)
I	I	I
I	0	I
I	I	I
I	0	I
I	I	I
I	0	I
I	I	I
I	I	I

Figure 5.15. Gibbs Sampling for Node Number 2.

Next, for node number 3, *CR (BFS, GT)* (as in Figure 5.11), the same steps were repeated to estimate the maximum degree of probability of the fact's validity that learning a BFS concept is necessary to know in order to reach the creation cognitive level to Create a Graph Traverse.

Figure 5.16 illustrates the estimation probability that is needed to be computed for the green node. The Markov blanket of this node consists of node1, node2, node 5, and node7 (the observations of which are illustrated in Figure 5.16). As seen in Figure 5.16, the node had neighbors in different cliques, so the sampling began with node number 3, where all other variables were temporally fixed with the same value. It is clear that node 7 is evidence that the value of this node is always true no matter which neighbor it has. By applying Equation 5.4:

$$P(x|MB(x))=\frac{exp(\sum_i s_i f_i(x))}{exp(\sum_i s_i f_i(x=0))+exp(\sum_i s_i f_i(x=1))}$$

First, the **Numerator** in Equation 5.4 was calculated, and

$exp(\sum_i s_i f_i(x))$ counts all the assignments where *CR (BFS, GT)* is True. This is based on the generated sampling using Gibbs sampling. Sample *CR (BFS, GT)* given its Markov blanket, and repeat. Count the number of times that *CR (BFS, GT)* is true and false in the samples. As illustrated in Figure 5.16, the sampling is generated randomly, where *CR (BFS, GT)* is equal to:

$$exp\left(\sum_i s_i f_i(x)\right) = (5*e^{12S}) = 5e^{12S}$$

The **Denominator** in Equation 5.4. is $exp\left(\sum_i s_i f_i(x=0)\right) + exp(\sum_i s_i f_i(x=1))$

In Equation 5.4 $exp\left(\sum_i s_i f_i(x=0)\right)$ represents the cases where *CR (BFS, GT)* is False, conditional of its *MB* (their first level neighbor) being False too. That means

P (CR (BFS, GT) = 0 |CR (Graph, GT) = 0, AP (Graph, GT) = 0, AP (Graph, BFS) = 0, CR (GT, Graph) = 0)).

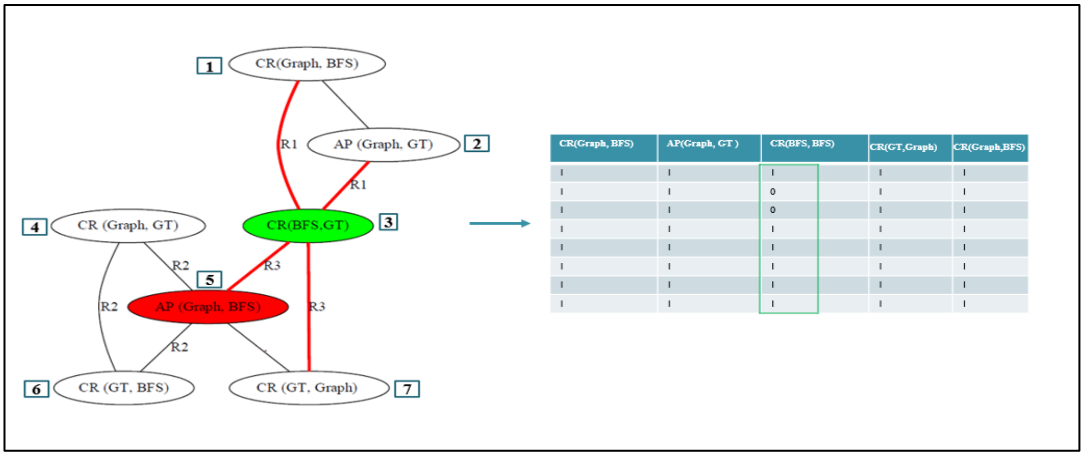
In Equation 5.4, $exp\left(\sum_i w_i f_i(x=1)\right)$ represents the cases where *CR (BFS, GT)* is True, conditional of its *MB* being True too. That means

P (CR (BFS, GT) =1 |CR (Graph, GT) =1, AP (Graph, GT) =1, AP (Graph, BFS) =1, CR (GT, Graph) =1))

$$= 5*e^{12S} = 5e^{12S}$$

Then, by substituting in Equation 5.4, the maximum probability of the fact of learning Graph to apply GT is the following:

$$P\ (AP\ (Graph,\ GT) = \frac{6e^{12}}{2+6e^{12}} \qquad (5.3.11)$$



CR(Graph, BFS)	AP(Graph, GT)	CR(BFS, BFS)	CR(GT,Graph)	CR(Graph,BFS)
I	I	I	I	I
I	I	0	I	I
I	I	0	I	I
I	I	I	I	I
I	I	I	I	I
I	I	I	I	I
I	I	I	I	I
I	I	I	I	I

Figure 5.16. Gibbs Sampling for Node Number 3.

The sampling procedure iteratively drew samples from the full conditional distributions for the rest of nodes in the graph (node 4, node 6, and node 7) except for the evidence node, which is the red node in Figure 5.16. After finishing the iteration over all the nodes, one cycle was completed. Then, the new sampling cycle started. The Gibbs sampling procedure was performed until convergence was reached.

Table 5.16. The Probability of the Fact at each Iteration.

Inferred Facts	Iteration-1	Iteration-2	Iteration-3	Iteration-4	Iteration-10
CREATE (Graph, BFS)	0.497000	0.535000	0.502941	0.504444	**0.500476**
APPLY (Graph, GT)	0.483333	0.500000	0.506667	0.501176	**0.514167**
CREATE (BFS, GT)	0.502407	0.502482	0.5902482	0.692411	**0.7976**

CREATE (Graph, GT)	0.499009	0.497163	0.497163	0.4956	**0.560000**
APPLY (Graph, BFS)	1	1	1	1	1
CREATE (GT, BFS)	0.483333	0.482222	0.484167	0.490000	**0.501154**
CREATE (GT, Graph)	0.520000	0.520000	0.528333	0.502067	**0.502067**

Ten iterations were used in this example, as Table 5.17 illustrates the values of the fact in each iteration. Through this example, it is clear that the value of the evidence, the raw number six, was equal to one in all iterations, meaning that they were fixed. Meanwhile, the other values for each fact depended on their values in the previous iteration; however, the sampling procedure was known to converge on the desired posterior distribution.

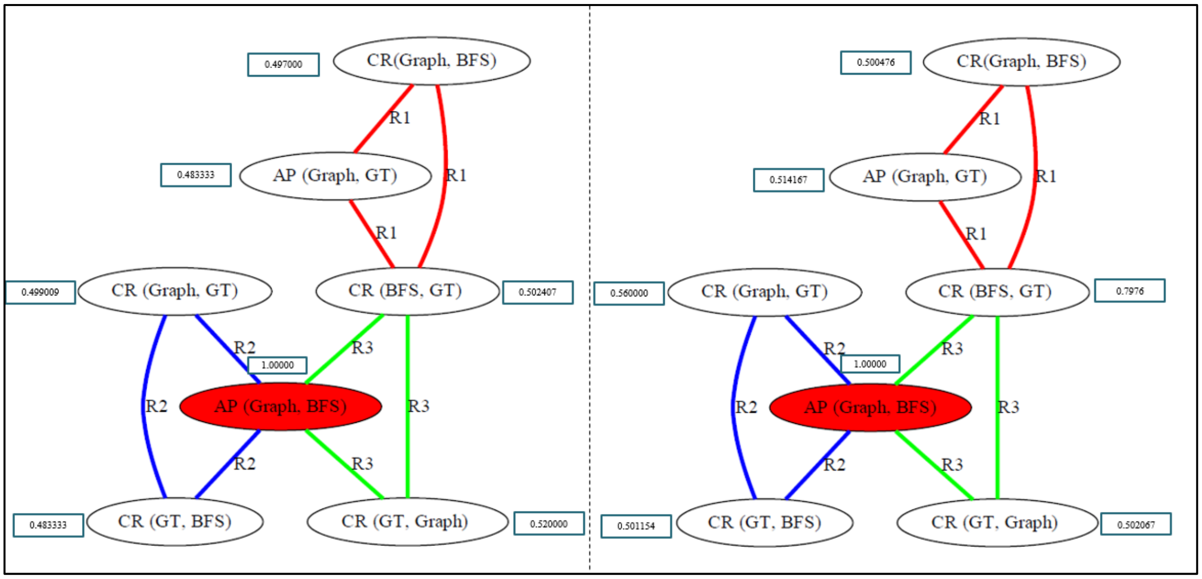


Figure 5.17. The Probability of the Fact at the First Iteration and Last Iteration.

5.6 Summary and Discussion

This chapter outlined and discussed the main concepts of the proposed methodology, a Markov Network (*MN*), a Markov Logic Network (*MLN*), and a Markov Cognitive Knowledge State Network (*MCKSN*). *MN* introduced the central concepts to understand both *MLN* and *MCKSN*.

MLN built a framework of combining the logic with the Markov Network. *MCKSN* used a framework to tackle the cognitive problem into a new context. The chapter also introduced the Markov Network with a practical example, as well as showing the intersection point of the Markov Network and Markov Logic Network which was generating the potential function based on the logic. Additionally, *MCKSN* introduced a methodology that showed promising results where the task of knowing concepts at a particular cognitive level was the focus. The *MCKSN* technique makes use of the cognitive domain in Computer Sciences.

The next chapter will discuss the experiment step and the evaluation of the Markov Cognitive Knowledge State Network (*MCKSN*).

CHAPTER 6: EXPERIMENT RESULTS AND EVALUATION

6.1 Introduction

This chapter presents an experimental validation on the *MCKSN* model's performance in identifying the hidden Cognitive Skill Dependencies (*CSD*) when compared to the results done by human identification. The chapter then illustrates the details of the used dataset, along with the design of the human model to assess the efficiency of the *MCKSN* model. Finally, it presents the comparison between the human model and the *MCKSN* model, as well as proof that the proposed model acts like humans when inferring the hidden Cognitive Skill Dependencies.

6.2 Test Dataset

This section discusses the bench mark dataset used in the experiments. An experiment was conducted on *Introduction to Algorithms,* a highly adapted textbook used in Computer Science classes at many universities. Table 6.1 provides a breakdown of the information about the chosen textbook.

Table 6.1. Statistical Information about the Textbook.

Algorithm Textbook	**Statistical Information**
Table of Content depth	4
Number of Sentences	11077
Number of Paragraphs	5959
Number of concepts (Nouns and Verbs)	207356
Number of Noun Concepts	2384
Number of Verb Concepts	354
Number of Extracted Relationships (Verb)	3886
Number of Extracted *CSBT* Relationships	615

Number of Extracted CSWN Relationships	680

Twenty-Two CS concepts were used in this experiment (see Table 6.2). The concepts were used as an input for both the human model and the *MCKSN* model. The experiment was only done for a subset of concepts, as the manual process of evaluating Cognitive Skill Dependencies is time consuming. The discussion led to the researcher choosing the concepts included in the course learning objectives.

Table 6.2. Several CS Concepts from the Algorithm Textbook.

CS- Concepts
1. ALGORITHM
2. DATA-STRUCTURE
3. BINARY-SEARCH-TREES
4. FLOYD-WARSHALL-ALGORITHM
5. GREEDY-ALGORITHM
6. RED-BLACK-TREES
7. LONGEST-COMMON-SUBSEQUENCE
8. SORTING-IN-LINEAR-TIME
9. BREADTH-FIRST-SEARCH
10. GRAPH-ALGORITHM
11. COUNTING-SORT
12. PRIM-ALGORITHM
13. LISTS
14. STACKS
15. RUNNING-TIME
16. WORST-CASE
17. HEAP-SORT-ALGORITHM

18. SORTING-ALGORITHM
19. QUICK-SORT-ALGORITHM
20. MST
21. TOPOLOGICAL-SORT
22. STRONGLY CONNECTED COMPONENT

Figure 6.1 illustrates a Semantic Knowledge Map (*SKM*) view of a fully connected graph for the twenty-two concepts. In the context of this experiment, initially the assumption is that there are Cognitive Skill Dependencies between any two concepts (nodes). In other words, the Semantic Knowledge Map (*SKM*) is a fully connected graph. Each two nodes in the graph have four types of dependencies. In reality, some of the nodes are not connected, while others are strongly connected based on their Cognitive Skill Dependencies. A fully connected graph was given to both the *MCKSN* model and the human model. Both models were then used to attempt to label the Cognitive Skill Dependencies between concepts in the graph.

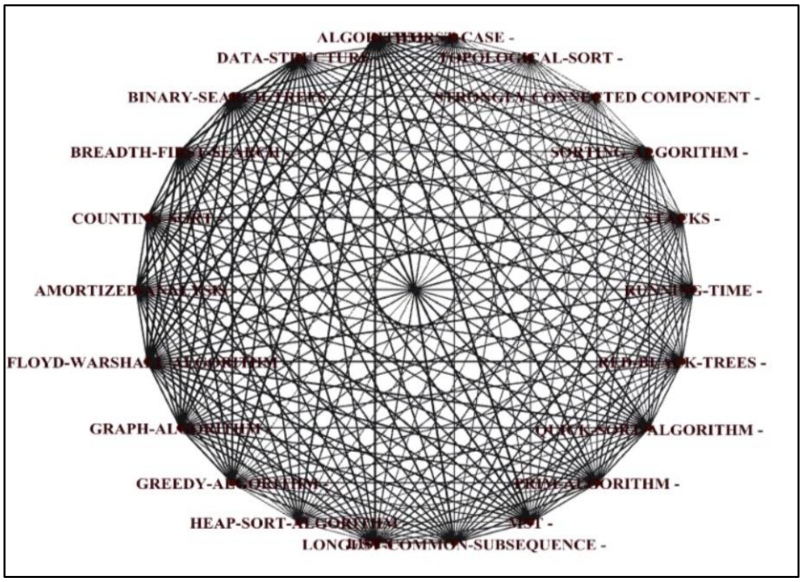


Figure 6.1. Semantic Knowledge Map (SKM) with the Cognitive Skill Dependencies between Concepts.

6.3 Skill Inference Rules (*SIR*) Extraction

The Skill Inference Rules (*SIR*) were generated in this experiment (Table 6.3 illustrates the generated *SIR*). The *SIR's* were used as input to guide the *MCKSN* model to infer the Cognitive Skill Dependencies. The generated *SIR's* are deliberately meant to be simple because if the *SIR* is too complex, it is possible for it to be neither valid nor not-valid. In other words, there would be a chance of generating a not-valid *SIR*. The SIR's in this experiment are generated by a group of Ph.D. students in the research phase with expertise in the Algorithm area. After many meetings and discussions, the *SIR's* were generated and evaluated. The group introduced their best efforts in this, taking into account the *SIR* format of the First Order Logic.

There is no universal method to determine whether a Skill Inference Rules (*SIR*) in First Order Logic is logically valid or not-valid, but the structure satisfies the validity of the *SIR*. In general, it's up to the cleverness and creativity of the experts to make a valid determination.

Table 6.3. Skill Inference Rules (SIR)

//Cognitive Skill Dependencies (**Understanding**) Rules
Understanding (*z*, *x*) ^ Understanding (*z*, *y*) => Understanding (*x*, *y*)
Understanding (*z*, *x*) ^ Applying (*z*, *y*) => Understanding (*x*, *y*)
Understanding (*z*, *x*) ^ Applying (*z*, *y*) => Applying (*x*, *y*)
Understanding (*z*, *x*) ^ Analyzing (*z*, *y*) => Understanding (*x*, *y*)
Understanding (*z*, *x*) ^ Analyzing (*z*, *y*) => Analyzing (*x*, *y*)
Understanding (*z*, *x*) ^ Creating (*z*, *y*) => Understanding (*x*, *y*)
Understanding (*z*, *x*) ^ Creating (*z*, *y*) => Creating (*x*, *y*)
//Cognitive Skill Dependencies (**Applying**) Rules
Applying (*z*, *x*) ^ Applying (*z*, *y*) => Applying (*x*, *y*)
Applying (*z*, *x*) ^ Understanding (*z*, *y*) => Applying (*x*, *y*)
Applying (*z*, *x)* ^ Understanding (*z*, *y*) => Understanding (*x*, *y*)
Applying (*z*, *x*) ^ Analyzing (*z*, *y*) => Applying (*x*, *y*)
Applying (*z*, *x*) ^ Analyzing (*z*, *y*) => Analyzing (*x*, *y*)
Applying (*z*, *x*) ^ Creating (*z*, *y*) => Applying (*x*, *y*)
Applying (*z*, *x*) ^ Creating (*z*, *y*) => Creating (*x*, *y*)
//Cognitive Skill Dependencies (**Analyze**) Rules
Analyze (*z*, *x*) ^ Analyze *(z, y)* => Analyze (*x*, *y*)
Analyze *(z*, *x*) ^ Understanding *(z, y)* => Analyze (*x*, *y*)
Analyze (*z*, *x*) ^ Understanding *(z, y)* => Understanding (*x*, *y*)
Analyze (*z*, *x*) ^ Applying *(z, y)* => Analyze (*x*, *y*)
Analyze (*z*, *x*) ^ Applying *(z, y)* => Applying (*x*, *y*)
Analyze (*z*, *x*) ^ Creating *(z, y)* => Analyze (*x*, *y*)
Analyze (*z*, *x*) ^ Creating *(z, y)* => Creating (*x*, *y*)
//Cognitive Skill Dependencies (**Creating**) Rules
Creating (*z*, *x*) ^ Creating *(z, y)* => Creating (*x*, *y*)
Creating (*z*, *x*) ^ Understanding *(z, y)* => Creating (*x*, *y*)
Creating (*z*, *x*) ^ Understanding (*z*, *y*)) => Understanding (*x*, *y*)
Creating (*z*, *x*) ^ Analyzing *(z, y)* => Creating (*x*, *y*)
Creating (*z*, *x)* ^ Analyzing *(z, y)* => Analyzing (*x*, *y*)
Creating (*z*, *x*) ^ Applying *(z, y)* => Creating (*x*, *y*)
Creating (*z*, *x*) ^ Applying *(z, y)* => Applying (*x*, *y*)

6.4 Human Evaluation Model Experiment

The purpose of the human evaluation was to measure the performance reliability of the *MCKSN* model. The ground truth was provided by two group of students since they have intuitive understanding about the Cognitive Skill Dependencies. The students were only asked to label the Cognitive Skill Dependencies between concepts for the final answer. Human judgment, the most widely accepted form of judgment, was used in order to best evaluate the proposed model. The human model experiment will be explained in detail in Section 6.3 in this chapter.

6.4.1 Evaluation Procedure

Procedures for selecting the participants and collecting the data are described in this section. The first thing to describe here is The Data Collection. Data was collected on November 26th during the Fall 2016 Semester. The collecting of the data is intentionally done at the end of the semester because the learning of the chosen concepts has already been completed by then. This study employed one mode of data collection: a student survey. The survey was used to label the Cognitive Skill Dependencies between concepts. The survey contained multiple-choice questions. There are four types of Cognitive Skill Dependencies between concepts: *Understanding*, *Analyzing*, *Applying-Evaluating*, and *Creating* (which are denoted as {BL_1, BL_2, BL_3, BL_4} respectively)(Nafa & Khan, 2015).

During the study, the researcher allocated the final 10 minutes of the class period to explain the survey procedure to the participants. The survey was a take home survey. A description of Bloom’s Taxonomy levels was given to participants with a simple example to explain the difference between each of the cognitive levels. Participants were told that the survey would not be graded and that their responses would not be shown to their instructor. The survey was given to participants, asking them questions based on their understanding and knowledge. Participants were also free to use external references to classify the Cognitive Skill Dependencies between each of the two concepts, where more than one type could be possible between two concepts.

Secondly, there were The Participants. The evaluation of the *MCKSN* model performance reliability focused on data purposefully obtained from a group of students. The students were deemed to be accurate learners to identify Cognitive Skill Dependencies because they had already learned the concepts used in this experiment prior to the experiment taking place. The study utilized two groups of co-ed students whose ages ranged from 23 to 55. The first group was composed of students enrolled in the "Design and Analysis of Algorithms" course offered by the Computer Science department at Kent State University. The second group consisted of Computer Science students at the masters and PhD levels who had already taken this course.

The class had 80 students, seventy of whom responded to participate in the study. Of the 80 students who responded to participate, ten students either subsequently declined to participate, or revealed that they were absent. As such, they were removed from consideration for this study. Consequently, the second sample consisted of 50 participants: Masters and PhD students who had taken this course previously.

The total participants for this study were 120 learners. Participants were orally informed that they could choose not to participate in the study. The class instructor was not present while the survey was administered so that students would not feel intimidated regarding their participation. As a group, these 120 study participants had the following characteristics: Seventy-five percent of the study's participants were male, and twenty-five percent were female. Ninety-two percent of the participants were CS majors, while eight percent were non-CS majors. The *GPA* for study participants varied in four different groups, where thirty percent had *A's*, thirty-eight percent had *B's*, twelve percent had *C's* and seventeen percent had *D's*. Table 6.4 illustrates the statistical information about the participants.

Table 6.4. Statistical Information about the Participants.

Gender		**Major**		**GPA**			
Female	Male	CS	Non-CS	A	B	C	D
25%	75%	92%	8%	30%	38%	12%	17%

Finally, there was the Institutional Review Board (IRB) Processing. This was the necessary approval given to conduct this study by Kent State University's Institutional Review Board (*IRB*). A reproduction of the e-mail message for the study's approval is provided in Appendix B. Rather than include a picture of the email printout, the e-mail message was typed out to preserve the anonymity of the institution at which this study was conducted. Once potential participants had been identified, instructions were given to them, along with the Informed Consent Form. It included a description of the research study, research procedures, risks and benefits of participation in the study, participant's rights, and protection of confidentiality. Students who signed the consent form became participants in the study. Before handing out the survey, the researcher asked participants if they had read and understood the consent form. Participants then received details about the process and procedures, along with a copy of the survey questions.

6.4.2 Human Model Result

The final data of the Human model were tabulated in eight tables, namely H_1, H_2, H_3, *and* H_4, respectively (as given in Appendix C). The Human model (H_b) tables contain Cognitive Skill Dependencies estimated by a human. For the evaluation, 460 Cognitive Skill Dependencies between 22 concepts were picked. Thus, in each table, there are 460 rows and 120 columns, where the rows denote the Cognitive Skill Dependencies between each two concepts and the columns represent human answers. The Human model was represented as a matrix H_b, where rows represent the ith Cognitive Skill Dependencies identified by the s^{th} human subject (columns) for each Cognitive Skill Dependency, and where cell values are human agreement counts for Cognitive Skill Dependencies. The Human model matrix Hb can be defined as follows:

$$H_b[i, s]$$

Where:

i: is the Cognitive Skill Dependencies index $1 \leq i \leq 460$, **s**: is the human subject index $1 \leq s \leq 120$, and **b**: is the Cognitive Skill Dependency index $1 \leq b \leq 4$. Then the Human-Evaluation matrix $H_b[i, s]$ was converted to a vector of elements $H_b[i]$.

6.5 A Markov Cognitive Knowledge State Network (*MCKSN*) Model Experiment

This section handles the experiments of applying the *MCKSN* model with a different set of *SIR's* as explained in section 6.2 in this chapter. The implementation of the *MCKSN* model is based on using *pracMLN*, an *MLN* python package for statistical relational learning and reasoning (Ankan & Panda, 2015). An input for this model (as mentioned in Chapter 5) consists of a Semantic Knowledge Map (*SKM*), a set of logic Inferential Algebra, facts, and CS Concepts. The *MCKSN* model is queried for four different Cognitive Skill Dependencies (BL_1, BL_2, BL_3, *and* BL_4), which are Understanding, Applying, Evaluating-Analyzing and Creating, respectively.

As a final result of the *MCKSN* model, four databases were created (BL_1, BL_2, BL_3, *and* BL_4, respectively). Each column in the database represents a percentage of the probability of the Cognitive Skill Dependencies between the concepts. The *MCKSN* model results are called (M_b). There are also expanded graphs in (Appendix A) of the dissertation. These present the Cognitive Skill Dependencies between the concepts based on their degree probability where the graphs are plotted according to a specific threshold (β_m). In these experiments, β_m had four different values (0.50, 0.65, 0.75, and 0.85, respectively) for each Bloom level (BL_1, BL_2, BL_3, *and* BL_4).

6.6 Comparing *MCKSN* Model and Human Model of inferring Cognitive Skill Dependencies.

The results of the *MCKSN* model were tabulated in eight tables, namely BL_1, BL_2, BL_3, *and* BL_4 respectively. The BL_1, BL_2, BL_3, BL_4 tables include Cognitive Skill Dependencies estimated by an *MCKSN* model (M_b). For the evaluation, 460 relationships between 22 concepts were picked. Thus, in each table, there are 460 rows and one column. The *MCKSN* model M_b can be defined as a vector of elements as follows:

$$M_b[i]$$

Where:

i: is the Cognitive Skill Dependencies index $1 \leq i \leq 460$ and ***b:*** is the Bloom level index $1 \leq b \leq 4$.

6.6.1 Data Preprocessing

The preprocessing step is used as an essential step of the compression procedure. In this study, the preprocessing steps are scaling, matching and mismatching, and performance matrix.

6.6.1.1 Scaling

Scaling is needed as a preprocessing step for computing the accuracy of the *MCKSN* model results $M_b[i]$ to better compare those with the Human-Evaluation $H_b[i]$ of discovering the Cognitive Skill Dependencies between concepts. In this context, scaling entails mapping all values to the same range. There is primary reason for doing this. Having all data in the same range eliminates the possibility of data with greater values dominating in the result and thus having a larger influence during the process of discovering the Cognitive Skill Dependencies. There are different techniques to scale the data (Muller & Guido, 2017; Müller & Guido, 2016). This section introduces the most widely used scaling techniques: Minmax and Log scaling techniques. The main reason for choosing them was to keep the dimensions of the data simple and convenient, thus matching the units.

Minmax Scaler, The Minmax Scaler is the most widely-used scaler, especially when dealing with the issue of classification. In the Minmax Scaler algorithm, the scaling was applied for both $H_b[i]$ and $M_b[i]$. The Minimax Scaler algorithm maps each contain an element in both $H_b[i]$ and $M_b[i]$ to new scaled elements, $H_b^{mm}[i]$ and $M_b^{mm}[i]$ respectively, by using the following Formulas 6.2 and 6.3:

$$H_b^{mm}[i]=\frac{H_b[i]\text{-}min(H_b[i])}{max(H_b[i])\ \text{-}min(H_b[i])} \qquad (6.2)$$

Where:

$\mathrm{H_b[i]}$ is the Human-Evaluation data and $\mathrm{H_b^{mm}[i]}$ is the scaled data, $\mathrm{max(H_b[i])}$ is the maximum value in $H_b[i]$ and $\min(H_b[i])$ is the minimum value in $\mathrm{H_b[i]}$.

$$M_b^{mm}[i]=\frac{M_b[i]\text{-}min(M_b[i])}{max(M_b[i])\ \text{-}min(M_b[i])} \qquad (6.3)$$

Where:

$M_b[i]$ is the data estimated by $MCKSN$ model, $M_b^{mm}[i]$ is the scaled data; $max(M_b[i])$ is the maximum value in $M_b[i]$ and $\min(M_b[i])$ is the minimum value in $\mathrm{M_b[i]}$.

Log Scaling, the log (LG) scaling takes the log of each data point, where each data point is replaced by its natural log. The log can be valuable both for making patterns in the data more interpretable and for helping to meet the assumptions of the used threshold. The scaling is applied for both $H_b[i]$ and $M_b[i]$. It can be calculated by using the following formulas 6.4 and 6.5:

$$H_b^{LG}[i]=Log(H_b[i]) \qquad (6.5)$$

Where:

$H_b[i]$ is the Human-Evaluation data and $H_b^{LG}[i]$ is the scaled data; and

$$M_b^{LG}[i]=Log(M_b[i]) \qquad (6.4)$$

Where:
$M_b[i]$ is the data estimated by $MCKSN$ model $and M_b^{LG}[i]$ is the scaled data.

6.6.1.2 Matching and Mismatching

Each Human Evaluation $H_b[i]$ element should match with the *MCKSN* model $M_b[i]$ elements for each Cognitive Skill Dependency i. That means:

$$M_b^{mm}[i] \approx H_b^{mm}[i] \; and \; M_b^{LG}[i] \approx H_b^{LG}[i]$$

The matching indicates how likely it would be for the results estimated by the *MCKSN* model $M_b[i]$ to match with those of the Human-Evaluation $H_b[i]$. In this study, the matching value was different from Cognitive Skill Dependency to another to restrict the result to only those cognitive Skill Dependency believed to be highly likely to be correct, driving confidence up to meet the human match rate. It can be said that $M_b^{mm}[i]$ matches with $H_b^{mm}[i]$, and vice versa, and that $M_b^{LG}[i]$ matches with $H_b^{LG}[i]$, and vice versa. The matching decision is described as follows:

$$H'^{mm}_1[i] = \begin{cases} 1 & if \quad H_1^{mm}[i] \geq 0.60 \\ 0 & if \quad H_1^{mm}[i] < 0.60 \end{cases} \quad \text{and} \quad M'^{mm}_1[i] = \begin{cases} 1 & if \quad M_1^{mm} \geq 0.61 \\ 0 & if \quad M_1^{mm}[i] < 0.61 \end{cases}$$

$$H'^{LG}_1[i] = \begin{cases} 1 & if \quad H_1^{LG}[i] \geq 4.0 \\ 0 & if \quad H_1^{LG}[i] < 4.0 \end{cases} \quad \text{and} \quad M'^{LG}_1[i] = \begin{cases} 1 & if \quad M_1^{LG} \geq 4.1 \\ 0 & if \quad M_1^{LG}[i] < 4.1 \end{cases}$$

Where $M'^{mm}_1[i]$ and $M'^{LG}_1[i]$ are the scaled data estimated by an *MCKSN* model, and $H'^{mm}_1[i] \; and \; H'^{LG}_1[i]$ are the scaled data estimated by the Human for Cognitive Skill Dependency (Understanding). In this formula the $H_1^{mm}[i]$ should be greater than or equal to 0.60; if this is true, it can then be said that Cognitive Skill Dependency (Understanding) exists between two concepts. If $H_1^{mm}[i]$ is less than 0.60, it can be said that no Cognitive

Skill Dependency (Understanding) can be found between the two concepts. This works in a similar way for other formulas as well.

$$H'^{mm}_2[i] = \begin{cases} 1 & if \quad H^{mm}_2[i] \geq 0.50 \\ 0 & if \quad H^{mm}_2[i] < 0.50 \end{cases} \text{ and } M'^{mm}_2[i] = \begin{cases} 1 & if \quad M^{mm}_2 \geq 0.56 \\ 0 & if \quad M^{mm}_2[i] < 0.56 \end{cases}$$

$$H'^{LG}_2[i] = \begin{cases} 1 & if \quad H^{LG}_2[i] \geq 4.0 \\ 0 & if \quad H^{LG}_2[i] < 4.0 \end{cases} \text{ and } M'^{LG}_2[i] = \begin{cases} 1 & if \quad M^{LG}_2 \geq 4.0 \\ 0 & if \quad M^{LG}_2[i] < 4.0 \end{cases}$$

Where $M'^{mm}_2[i]$ and $M'^{LG}_2[i]$ are the scaled data estimated by *MCKSN and* $H'^{mm}_2[i]$ and $H_2^{\prime LG}[i]$ are the scaled data evaluated by the Human subject for Cognitive Skill Dependency (Applying).

$$H'^{mm}_3[i] = \begin{cases} 1 & if \quad H^{mm}_3[i] \geq 0.60 \\ 0 & if \quad H^{mm}_3[i] < 0.60 \end{cases} \text{ and } M'^{mm}_3[i] = \begin{cases} 1 & if \quad M^{mm}_3 \geq 0.61 \\ 0 & if \quad M^{mm}_3[i] < 0.61 \end{cases}$$

$$H'^{LG}_3[i] = \begin{cases} 1 & if \quad H^{LG}_3[i] \geq 4.0 \\ 0 & if \quad H^{LG}_3[i] < 4.0 \end{cases} \text{ and } M'^{LG}_3[i] = \begin{cases} 1 & if \quad M^{LG}_3 \geq 4.1 \\ 0 & if \quad M^{LG}_3[i] < 4.1 \end{cases}$$

Where $M'^{mm}_3[i]$ and $M'^{LG}_3[i]$ are the scaled data estimated by *MCKSN* and $H'^{mm}_3[i]$ and $H_3^{\prime LG}[i]$ are the scaled data evaluated by the Human subject for Cognitive Skill Dependency (Analyzing).

$$H'^{mm}_4[i] = \begin{cases} 1 & if \quad H^{mm}_4[i] \geq 0.60 \\ 0 & if \quad H^{mm}_4[i] < 0.60 \end{cases} \text{ and } M'^{mm}_4[i] = \begin{cases} 1 & if \quad M^{mm}_4 \geq 0.62 \\ 0 & if \quad M^{mm}_4[i] < 0.62 \end{cases}$$

$$H'^{LG}_4[i] = \begin{cases} 1 & if \quad H^{LG}_3[i] \geq 4.0 \\ 0 & if \quad H^{LG}_3[i] < 4.0 \end{cases} \text{ and } M'^{LG}_4[i] = \begin{cases} 1 & if \quad M^{LG}_4 \geq 4.1 \\ 0 & if \quad M^{LG}_4[i] < 4.1 \end{cases}$$

Where $M'^{mm}_4[i]$ and $M'^{LG}_4[i]$ are the scaled data estimated by *MCKSN* and $H'^{mm}_4[i]$ and $H_4^{\prime LG}[i]$ are the scaled data evaluated by the Human subject for Cognitive

Skill Dependency (Creating). The following section introduces the Performance matrix used to identify the evaluation parameters and the types of errors found in the *MCKSN* model.

6.6.2 Performance Evaluation

The performance evaluation matrix is a tabulation of the performance of the *MCKSN* model of inferring Cognitive Skill Dependencies. Table 6.4 defines the most common performance evaluation matrix-based evaluation measures used in the literature (Davis & Goadrich, 2006; Fawcett, 2006). It relates the human evaluation number of the Cognitive Skill Dependency per class (as its rows) to the *MCKSN* number of Cognitive Skill Dependency per class (as its columns). The numeric values of the matrix generated during the comparison of human model and *MCKSN* model. Since there are two different scaling techniques, two different Performance matrixes were defined. The first performance matrix used was $A_{i,j}$, which contains the elements of $H'^{mm}_{b}[i]$ and with $M'^{mm}_{b}[i]$ as an entry. The second performance matrix is $B_{i,j,}$ which contains the elements of $H'^{LG}_{b}[i]$, with $M'^{LG}_{b}[i]$ as an entry (as seen in Table 6.5.)

Table 6.5. Two Performance Matrixes.

$\boldsymbol{A_{ij}}$	$M'^{mm}_{b}[i]$	
$HE'^{mm}_{b}[i]$	Yes	No
Yes	$\boldsymbol{a_{11}}$	$\boldsymbol{a_{12}}$
No	$\boldsymbol{a_{21}}$	$\boldsymbol{a_{22}}$

$\boldsymbol{B_{ij}}$	$M'^{LG}_{b}[i]$	
$HE'^{LG}_{b}[i]$	Yes	No
Yes	$\boldsymbol{b_{11}}$	$\boldsymbol{b_{12}}$
No	$\boldsymbol{b_{21}}$	$\boldsymbol{b_{22}}$

Before going further, several important terms for both Performance matrixes explained in Table 6.5 need to be defined, and they are as follows:

True Positives: in the A_{ij} matrix, the true positive case is the cell (a_{11}) representing the total number cases that are correctly estimated by the *MCKSN* model as Cognitive Skill Dependencies. In other words, these are the total matches between the *MCKSN* model and human model that were positive $(H'^{mm}_{b}[i] = M'^{mm}_{b}[i] = 1)$. Mathematically, it can be calculated by using Equation 6.6:

$$a_{11} = \sum_{i=1}^{S} H'^{mm}_{b}[i] * M'^{mm}_{b}[i] \qquad (6.6)$$

In the B_{ij} matrix, the true positive case is the cell b_{11} and the decision (in this case between $M'^{LG}_{b}[i]\ and\ H'^{LG}_{b}[i],$ where $(H'^{LG}_{b}[i]=M'^{LG}_{b}[i]=1)$. It can be calculated by using Equation 6.7:

$$b_{11}=\sum_{i=1}^{S} (H'^{LG}_{b}[i] * M'^{LG}_{b}[i]) \qquad (6.7)$$

False Negative: in the A_{ij} matrix, the false negative case is the cell (a_{12}) representing the total number of cases that are incorrectly estimated by the *MCKSN* model as non-Cognitive Skill Dependency. In other words, it was the sum of the mismatches between both models, where the mismatches were positive $(\ H'^{mm}_{b}[i] = 1\ and\ M'^{mm}_{b}[i] = 0)$. Mathematically, it can be calculated by using Equation 6.8:

$$a_{12} = \sum_{i=1}^{S} [H'^{mm}_{b}[i] - (H'^{mm}_{b}[i] * M'^{mm}_{b}[i])] \qquad (6.8)$$

In the *B_{ij}* matrix, the false negative case is the cell b_{12} representing the decision (in this case between $M_b^{LG}[i]\ and\ H'^{LG}_b[i]$ where $H'^{LG}_b[i] = 1\ and\ M'^{LG}_b[i] = 0$). It can be calculated by using Equation 6.9:

$$b_{12} = \sum_{i=1}^{S} [H'^{LG}_b[i] - (H'^{LG}_b[i] * M'^{LG}_b[i])] \qquad (6.9)$$

False Positive: in the A_{ij} matrix, the false negative case is the cell (a_{21}) representing the total number of cases that are incorrectly estimated by the *MCKSN* model as Cognitive Skill Dependency (Also known as a "Type II error"). In other words, it was the sum of the mismatches between both models, where the mismatches were negative ($\mathrm{H}'^{LG}_{\mathrm{b}}[\mathrm{i}] = 0$ and $\mathrm{M}'^{\mathrm{LG}}_{\mathrm{b}}[\mathrm{i}] = 1$). Mathematically, it can be calculated by using Equation 6.10:

$$a_{21} = \sum_{i=1}^{S} [M'^{mm}_b[i] - (H'^{mm}_b[i] * M'^{mm}_b[i])] \qquad (6.10)$$

In the B_{ij} matrix, the false positive case is the cell b_{21}, and the decision in this case is between $\mathrm{M}^{\mathrm{LG}}_{\mathrm{b}}[\mathrm{i}]$ and $\mathrm{H}'^{\mathrm{LG}}_{\mathrm{b}}[\mathrm{i}]$, where ($H'^{LG}_b[i] = 0\ and\ M'^{LG}_b = 1$). *It* can be calculated by using Equation 6.11:

$$b_{21} = \sum_{i=1}^{S} [M'^{LG}_b[i] - (H'^{LG}_b[i] * M'^{LG}_b[i])] \qquad (6.11)$$

True Negative: in the A_{ij} matrix, the false negative case is the cell (a_{22}) representing the total number of cases that are correctly estimated by the *MCKSN* model as non-Cognitive Skill Dependency. In other words, it was the sum of the matches between both models where the matches were negative ($M'^{mm}_b[i] = H'^{mm}_b[i] = 0$). Mathematically, it can be calculated by using Equation 6.12:

$$a_{22}=N-\sum_{i=1}^{S}[(H'^{mm}_{b}[i]+M'^{mm}_{b}[i])-(H'^{mm}_{b}[i]*M'^{mm}_{b}[i])] \quad (6.12)$$

In the B_{ij} matrix, the true negative case is the cell b_{22}, and the decision in this case is between $M'^{LG}_{b}[i]\ and\ H'^{LG}_{b}[i]$, where $M'^{LG}_{b}[i]=H'^{LG}_{b}[i]\ =0$. It can be calculated by using Equation 6.13:

$$b_{22}= S-\sum_{i=1}^{s}\left[(H'^{LG}_{b}[i]+M'^{LG}_{b}[i])-(H'^{LG}_{b}[i]*M'^{LG}_{b}[i])\right] \quad (6.13)$$

Precision: this is a factor to measure how much of the *MCKSN* model's guess was correct (Grishman & Sundheim, 1996). Based on the first scaling technique, precision was calculated as follows in Equation 6.14:

$$P=\frac{a_{11}}{\sum_{i=1}^{2}a_{i1}} \quad (6.14)$$

Based on the second scaling technique, precision was calculated as follows in Equation 6.15:

$$P'=\frac{b_{11}}{\sum_{i=1}^{2}b_{i1}} \quad (6.15)$$

Recall: this is used as a performance metric when the *MCKSN* model predicts yes, and it is actually correct (Grishman & Sundheim, 1996). Based on the first scaling technique, the formula for calculating recall is given in Equation 6.16:

$$R=\frac{a_{1,1}}{\sum_{j=1}^{2}a_{1,j}} \quad (6.16)$$

Based on the second scaling technique, recall is calculated as follows in Equation 6.18:

$$R'=\frac{b_{11}}{\sum_{j=1}^{2} b_{1j}} \tag{6.18}$$

Accuracy: it is most commonly defined over all the classification errors that are made (Powers, 2011); therefore, based on the first scaling technique denoted as *D*, it is calculated as follows in Equation 6.19:

$$D=\frac{a_{11}+a_{22}}{\sum_{i=1}^{2}\sum_{j=1}^{2} a_{ij}} \tag{6.19}$$

Based on the second scaling technique, an accuracy *D'* is calculated as follows in Equation 6.20:

$$D' = \frac{b_{11} + b_{22}}{\sum_{i=1}^{2}\sum_{j=1}^{2} b_{ij}} \tag{6.20}$$

F-measure: this is a measure that uses both the precision and recall when testing accuracy. The calculation considers both *P* and *R* (Lewis, 1995). The *F-measure* score is at its highest point when it equals 1, and at its lowest when it equals 0. Initially introduced by van Rijsbergen(Van Rijsbergen, 1974), *F-measures* work as an evaluation criterion as follows:

$$F\cdot\ Measure = \frac{2}{\frac{1}{R}+\frac{1}{P}} = 2.\ \frac{P.R}{P+R} \tag{6.21}$$

Based on the second scaling technique, the F'-measure is calculated as follows in Equation 6.22:

$$F'\cdot\ Measure = \frac{2}{\frac{1}{R'} + \frac{1}{P'}} = 2.\ \frac{P'.R'}{P'+R'} \qquad (6.22)$$

As a result of calculating the Performance matrix for Cognitive Skill Dependency (Understanding) using both scaling techniques illustrated in table 6.6.

Table 6.6. Two Performance Matrixes for Cognitive Skill Dependency (Understanding).

***A**$_{ij}$*	$M'^{mm}_{1}[i]$	
$HE'^{mm}_{1}[i]$	Yes	No
Yes	**261**	**64**
No	**8**	**127**

***B**$_{ij}$*	$M'^{LG}_{1}[i]$	
$HE'^{LG}_{1}[i]$	Yes	No
Yes	**336**	**2**
No	**68**	**119**

Table 6.6 presents the eight instance values calculated from performance matrix of the four entries that are presented in the table for both scaling techniques (MinMax scaling and Log scaling) for the cognitive skill dependency (Understanding).

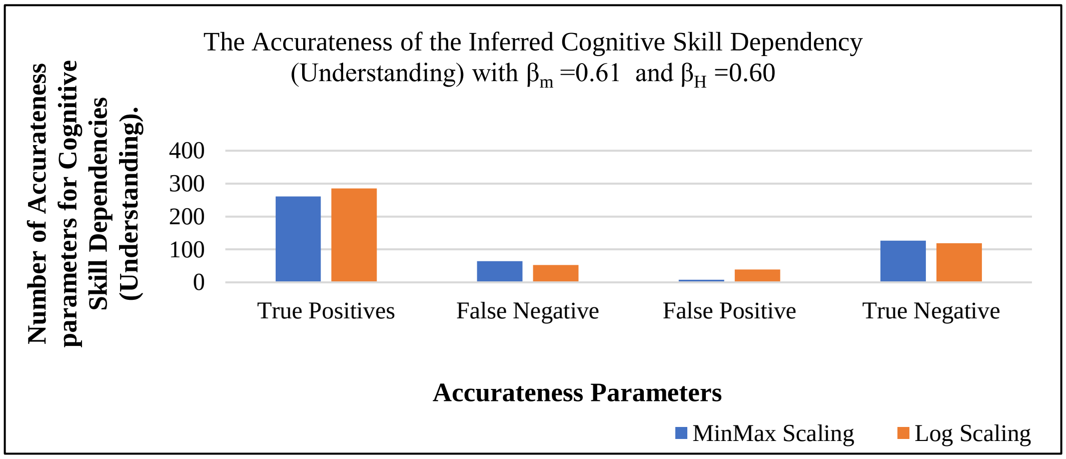


Figure 6.2. The Accurateness of the Inferred Cognitive Skill Dependencies (Understanding).

Figure 6.2 has been shown that the True Positive (*TP*) rate measures the fraction of positive Cognitive Skill Dependencies that are correctly labeled as positive. The True Negative (*TN*) rate measures the fraction of positive Cognitive Skill Dependencies that are correctly labeled as negative. In the other hand, False Positive (*FP*) and False Negative (*FN*) measures the fraction of cases that are misclassified of Cognitive Skill Dependencies. It is clear that, *TP* and *FN* in both scaling techniques shows the highest values in the Figure. It means that the *MCKSN* model is suitable for the research problem and makes well behavior to estimate the Cognitive Skill Dependency (Understanding). Meanwhile, the pattern of the misclassification rates in the False Positive (*FP*) and False Negative (*FN*) were low. Overall, the higher the number of the accurateness parameters (*TP and TN*), the more correct the estimation of Cognitive Skill Dependencies; the reverse occurred with respect to incorrect classification.

Table 6.7. Two Performance Matrixes for Cognitive Skill Dependency (Applying).

$\boldsymbol{A_{ij}}$	$M'^{mm}_{2}[i]$	
$HE'^{mm}_{2}[i]$	Yes	No
Yes	**324**	**49**
No	**52**	**36**

$\boldsymbol{B_{ij}}$	$M'^{LG}_{2}[i]$	
$HE'^{LG}_{2}[i]$	Yes	No
Yes	**301**	**23**
No	**76**	**60**

Table 6.7 presents the eight instance values calculated from the performance matrix of the four entries presented in the table for both scaling techniques (MinMax scaling and Log scaling) for the cognitive skill dependencies (Applying).

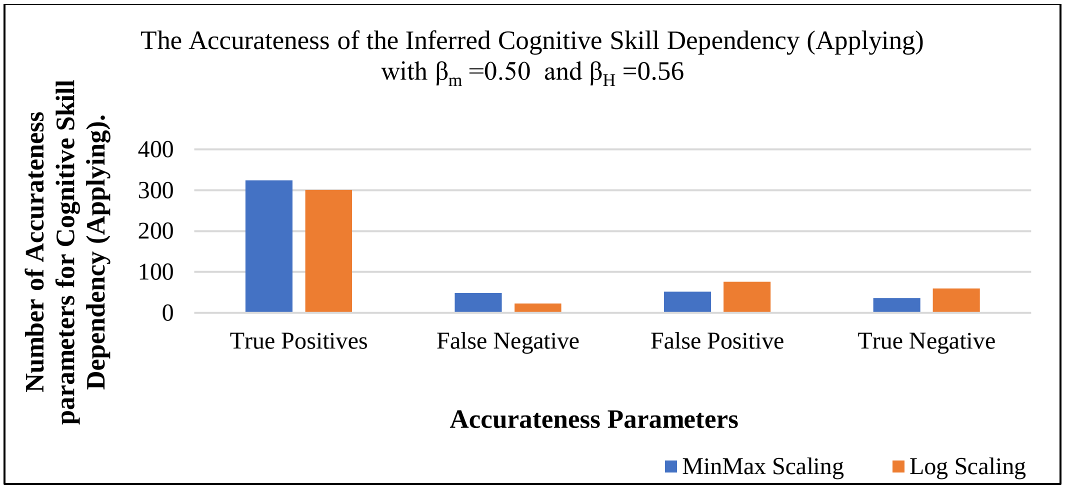


Figure 6.3. The Accurateness of the Inferred Cognitive Skill Dependencies (Applying).

Figure 6.3 has been shown the accurateness parameters used to gauge the performance of the *MCKSN* model for Cognitive Skill Dependencies (Applying). The Figure illustrated that True Positive (*TP*) and True Negative (*TN*) rates in both scaling techniques shows the pick values in the Figure. while keeping the False Positive (*FP*) and False Negative (*FN*) rates at an acceptable level. Clearly, the *MCKSN* model generates only a small number of *FP*. The *MCKSN* model provides a meaningful estimation of the Cognitive Skill Dependency (Applying).

Table 6.8. Two Performance Matrixes for Cognitive Skill Dependency (Analyzing-Evaluating).

A A_{ij}	$M'^{mm}_3[i]$	
$HE'^{mm}_3[i]$	Yes	No
Yes	**322**	**37**
No	**42**	**59**

B_{ij}	$M'^{LG}_3[i]$	
$HE'^{LG}_3[i]$	Yes	No
Yes	**314**	**17**
No	**50**	**79**

Table 6.8 presents the eight instance values calculated from the Performance matrix of the four entries presented in the table for both scaling techniques (MinMax scaling and Log scaling) for the cognitive skill dependencies (Analyzing-Evaluating).

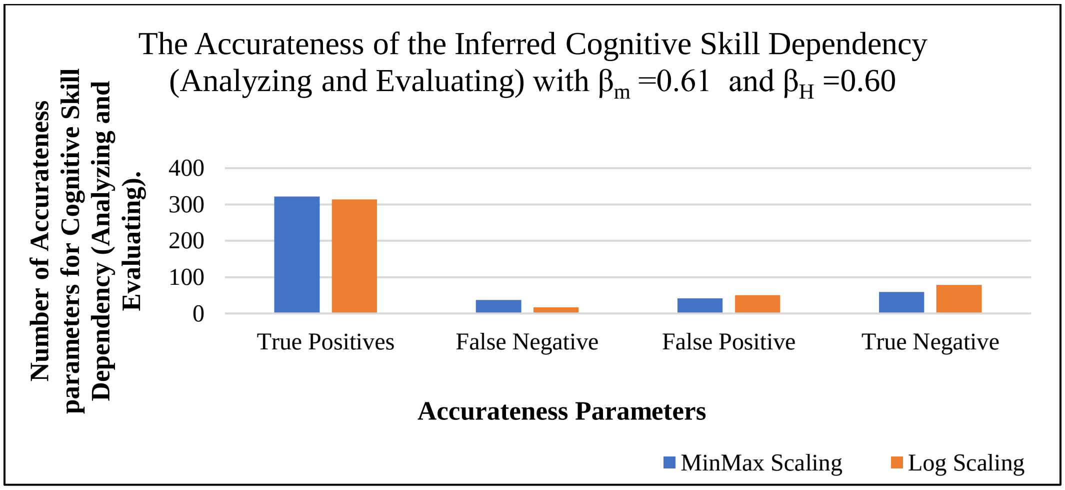


Figure 6.4. The Accurateness of the Inferred Cognitive Skill Dependencies (Analyzing-Evaluating).

Figure 6.4 has been illustrated that the True Positive (*TP*) and True Negative (*TN*) rates in both scaling techniques displays the highest values in the Figure. It means that the *MCKSN* model is suitable for the proposed problem and makes well behavior to estimate the Cognitive Skill Dependencies compared with a human subject. Meanwhile, the pattern of the misclassification rates in the False Positive (*FP*) and False Negative (*FN*) were low. Overall, the higher the number of the accurateness parameters (*TP and TN*), the more correct the estimation of Cognitive Skill Dependency; the reverse occurred with respect to incorrect classification.

Table 6.9. Two Performance Matrixes for Cognitive Skill Dependencies (Applying).

A_{ij}	$M'^{mm}_{2}[i]$	
$\mathrm{HE}'^{\mathrm{mm}}_{2}[\mathrm{i}]$	Yes	No
Yes	**324**	**49**
No	**52**	**36**

$\boldsymbol{B_{ij}}$	$M'^{LG}_{2}[i]$	
$\mathrm{HE}'^{\mathrm{LG}}_{2}[\mathrm{i}]$	Yes	No
Yes	**301**	**23**
No	**76**	**60**

Table 6.9 presents the eight instance values calculated from the Performance matrix of the four entries presented in the table for both scaling techniques (MinMax scaling and Log scaling) for the cognitive skill dependencies (Applying).

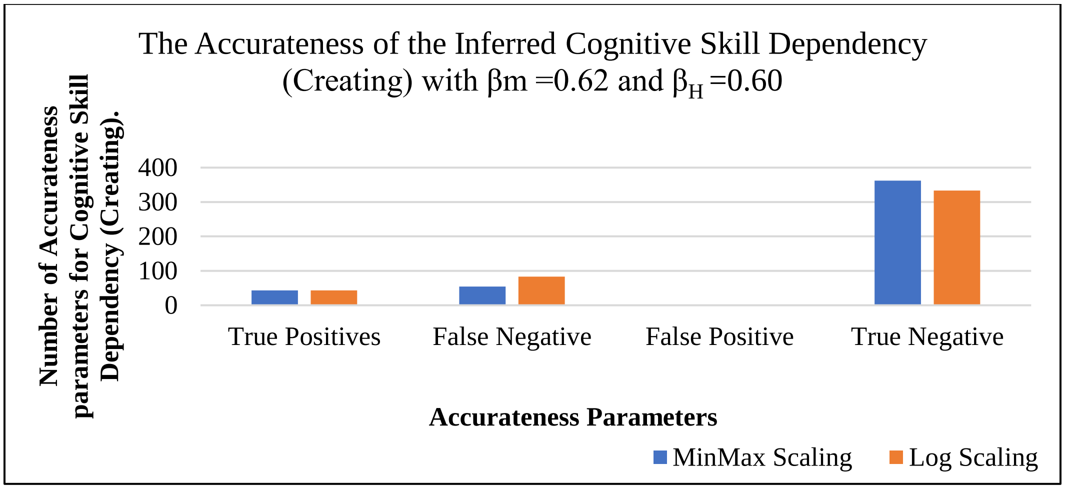


Figure 6.5. The Accurateness of the Inferred Cognitive Skill Dependencies (Creating).

In Figure 6.5, it is clear that the True Negative (*TN*) rate is a significantly higher than those of the other parameters, where the *TP*, *FN, and FP* are the lowest values. Overall, the result shows different performances for Inferred Cognitive Skill Dependencies

(Creating). This skill level is the highest Bloom skill, so it is rigid to estimate the Cognitive Skill Dependencies with high accuracy.

Overall, it is clear that the results show mostly similar behaviors for the *TP, TN*, *FP, and FN* estimated by the *MCKSN* model. From the obtained results, it is important to notice that the model behavior slightly differs from the inferring of Cognitive Skill Dependency (Creating). This is affected in a different way by the sample size, which is small compared to the others found in other Cognitive Skill Dependencies. To summarize, the *MCKSN* model can be used to obtain an optimal inference of the Cognitive Skill Dependencies, finding that the behavior of the model was very good. The analysis is supported by experimental results, showing the potential and practical use of the *MCKSN* model. There are other important properties and experiments to consider, making it interesting to further study the proposed model.

It is important to decide the usefulness of the *MCKSN* model. It has been shown that the task of assessing model performance is not trivial, and that there are many available evaluation measures to do so. Many of the performance measures are in some way derived from the Performance matrix, which enumerates the correct and incorrect predictions produced by the model.

Table 6.10 shows the performance of the Cognitive Skill Dependencies (Understanding), where all the measured values were obtained by the *MCKSN* model by using two different scaling techniques. The *MCKSN* model was successful in obtaining a Precision(*P*) of 97%, a Recall(*R*) of 66%, an accuracy (*D*)of 84%, and an *F-measure* of

78% for the MinMax scaling technique; and a Precision (*P'*) of 84%, a Recall(*R'*) of 84%, an accuracy(*D'*) of 81%, and an *F'-measure* of 84% for the Log scaling technique.

Table 6.10. Evaluation Parameters for Cognitive Skill Dependency (Understanding)

Evaluation Parameters							
P	***P'***	***R***	***R'***	***D***	***D'***	***F. Measure***	***F'. Measure***
0.97	0.84	0.66	0.84	0.84	0.81	0.78	0.84

As can be seen in Figure 6.13, the overall performance of the *MCKSN* model for Cognitive Skill Dependencies (Understanding) with respect to the evaluation parameters displayed high accuracy using both scaling techniques.

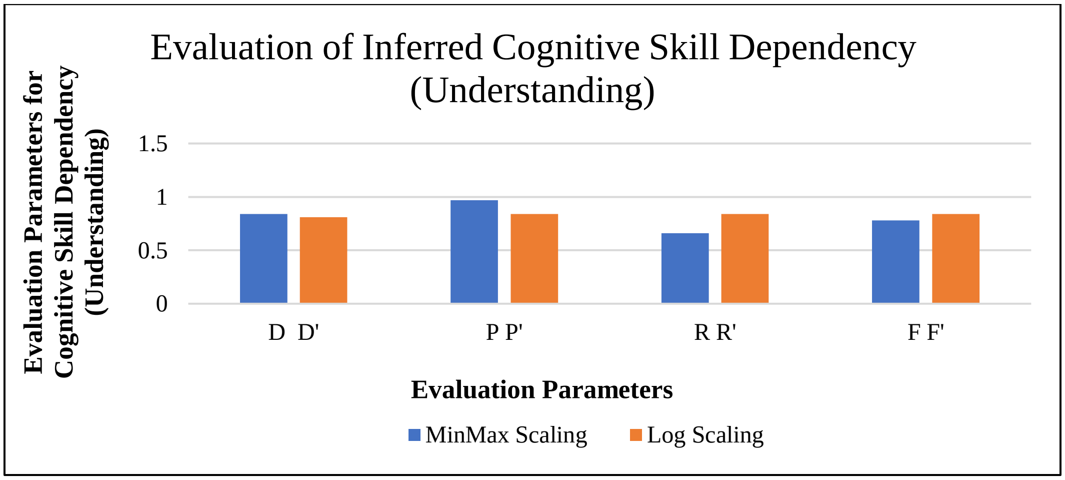


Figure 6.6. Evaluation parameters for Cognitive Skill Dependencies (Understanding).

Table 6.6 shows the performance of the Cognitive Skill Dependencies (Applying), where all the measured values were obtained by the *MCKSN* model using two different scaling techniques. The *MCKSN* model was successful in obtaining a Precision(*P*) of 81%, a Recall(R) of 100%, an accuracy(D) of 81%, and an F-measure of 86% for the MinMax scaling technique; and a Precision (P') of 95%, a Recall(R') of 100%, an accuracy(D') of 95%, and an *F'-measure* of 97% for the Log scaling technique.

Table 6.11. Evaluation Parameters for Cognitive Skill Dependency (Applying)

Evaluation Parameters							
P	***P'***	***R***	***R'***	***D***	***D'***	***F. Measure***	***F'. Measure***
0.81	0.95	1.0	1.0	0.81	0.95	0.86	0.97

As can be seen in Figure 6.7, the overall performance of the *MCKSN* model for Cognitive Skill Dependencies (Applying) with respect to performance evaluation parameters showed a high rate accuracy using both scaling technique.

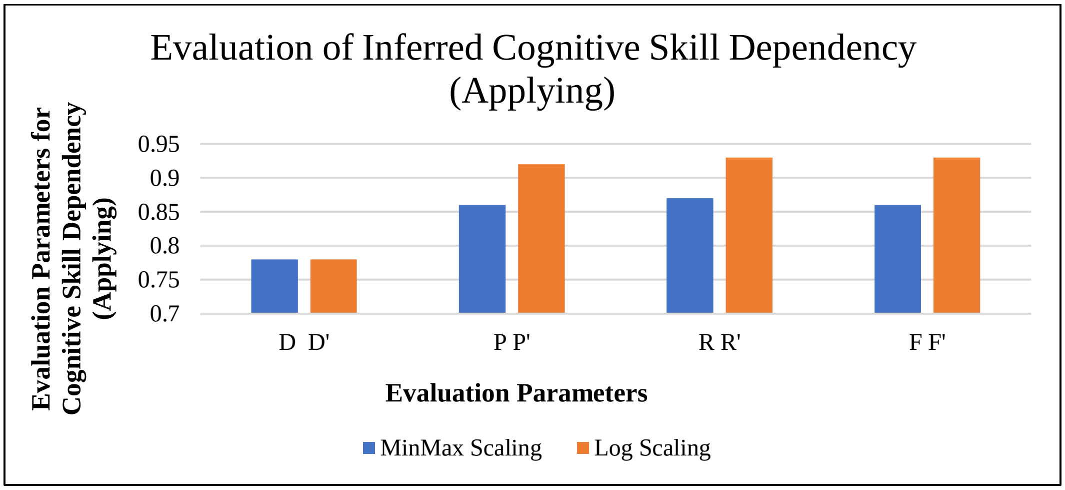


Figure 6.7. Evaluation parameters for Cognitive Skill Dependency (Applying).

Table 6.12 shows the performance of Cognitive Skill Dependencies (Analyzing-Evaluating), where all the measured values were obtained by the *MCKSN* model using two different scaling techniques. The *MCKSN* model was successful in obtaining a Precision(P)

of 88%, a Recall (R) of 88%, an accuracy(D) of 83%, and an *F-measure* of 89% for the MinMax scaling technique; and a Precision(P') of 95%, a Recall(R') of 95%, an accuracy(D') of 85%, and an F-measure of 95% for the Log scaling technique.

Table 6.12. Evaluation Parameters for Cognitive Skill Dependency (Analyzing-Evaluating)

Evaluation Parameters							
P	***P'***	***R***	***R'***	***D***	***D'***	***F. Measure***	***F'. Measure***
0.88	0.95	0.88	0.95	0.83	0.85	0.89	0.95

As can be seen in Figure 6.8, the overall performance of the *MCKSN* model for Cognitive Skill Dependency (Analyzing-Evaluating) with respect to performance evaluation parameters improved with the Log scaling technique.

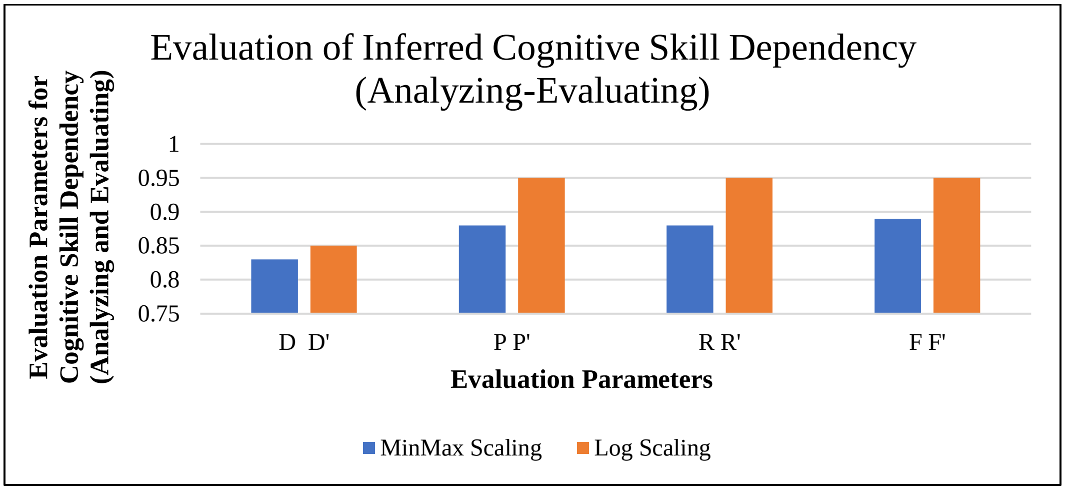


Figure 6.8. Evaluation parameters for Cognitive Skill Dependencies (Analyzing-Evaluating).

Table 6.13 shows the performance of the Cognitive Skill Dependencies (Analyzing-Evaluating). where all the measured values were obtained by the *MCKSN* model using two different scaling techniques. The *MCKSN* model was successful in obtaining a Precision(*P*) of 1%, a Recall(*R*) of 44%, an accuracy(*D*) of 88%, and *F-measure* of 17% for the MinMax scaling technique; and Precision (*P'*) of 34%, Recall (*R'*) of 34%, an accuracy(*D'*) of 81%, and *F'-measure* of 34% for the Log scaling technique.

Table 6.13. Evaluation Parameters for Cognitive Skill Dependency (Creating)

Evaluation Parameters							
P	***P'***	***R***	***R'***	***D***	***D'***	***F. Measure***	***F'. Measure***
0. 1	0.34	0.44	0.34	0.88	0.81	0.17	0.34

As can be seen in Figure 6.9, the overall performance of the *MCKSN* model for inferring Cognitive Skill Dependency (Creating) with respect to performance evaluation parameters

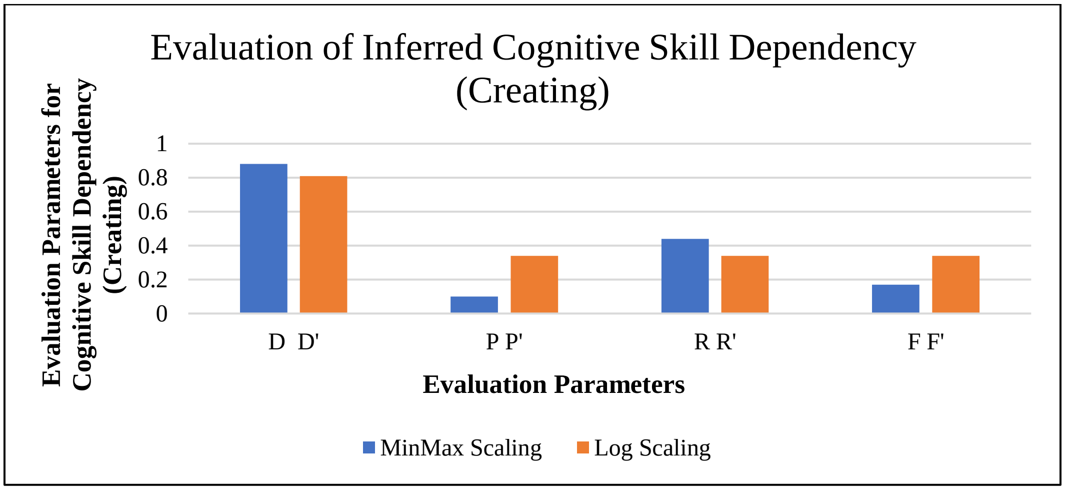


Figure 6.9. Evaluation parameters for Cognitive Skill Dependencies (Creating).

As illustrated in Figure 6.10, the accuracy rate for all the inferred Cognitive Skill Dependencies were promising - 84% ,78%,83%, and 88% for Understanding, Applying, Analyzing-Evaluating, and Creating, respectively.

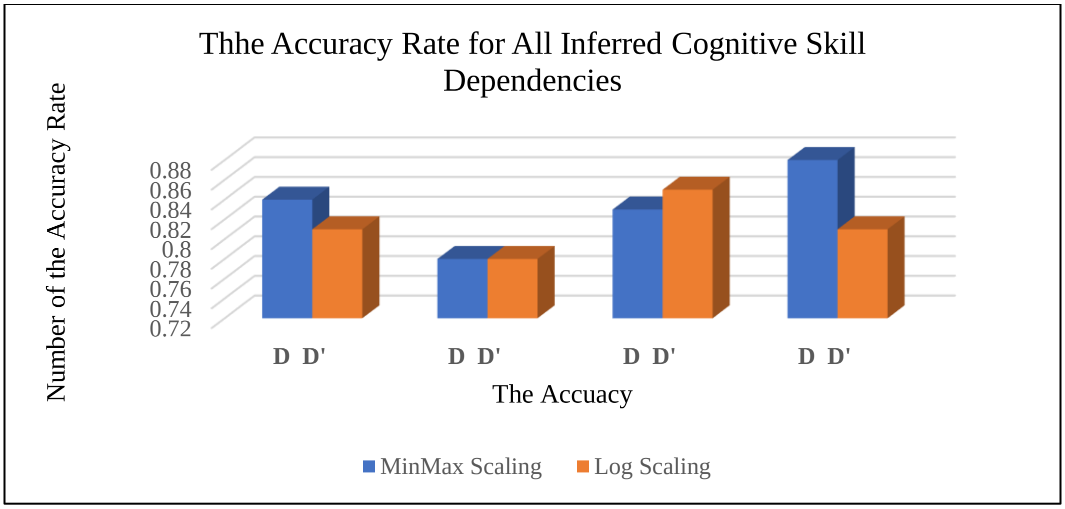


Figure 6.10. The Accuracy Rate for all the Inferred Cognitive Skill Dependencies.

6.7 Margin of Errors

Throughout this dissertation, the probability that the results of *MCKSN* model are significant means that the obtained results were accurate. Showing a high confidence level means that a very small probability of the *MCKSN* results happened by chance. The confidence level ranged from 0% to 100%. If the confidence level of the obtained results is zero, it means that there is no faith at all in getting the same results if the human experiments were to be repeated. If the confidence levels are less than 100, it means that there is no doubt at all that if the experiment were repeated that it would get the same results. The confidence level value (*CL*) is called the Wilson score interval (Gilbert, 1987)The values of the *CL* are provided from statistics, and common values used are: (90%, 95%, 98%, and 99%).

In *MCKSN* model experiments, the confidence level used to calculate the margin of errors was 95%. The Margin of Errors (*ME*) is the probability of any type of errors in the *MCKSN* model results. Knowing the margin of error for the result helps to estimate how close the *MCKSN* results are to the truth, based on the Human Evaluation.

The formula to calculate *ME* used in this dissertation was *Z* times the Standard Error as in 6.23.

Where:

$$ME = Z \text{ x } SE \tag{6.23}$$

***Z*:** z-score is the value that meets the *CL* = 95% (1.96 in the z-table). The z-table is a statistical table allowing to interpret the results marked in that table. It can tell what percentage is under the curve at any particular point (Brownlee, 1965)..

SE: is the standard error calculation that can be done by the mathematical formula 6.24:

$$SE = \sqrt{\frac{M'^{mm}_{b}[i]_e(M'^{mm}_{b}[i]_e - 1)}{N}} \tag{6.24}$$

Where:

N: is the sample size, and $M^{mm}_{b}[i]_e$ is an *MCKSN* error. It is calculated as in Equation 6.25.

$$M^{mm}_{b}[i]_e = \frac{a_{22}}{N} \tag{6.25}$$

$$a_{22} = N - \sum_{i=1}^{N} [(H'^{mm}_{b}[i] + M'^{mm}_{b}[i]) - (H'^{mm}_{b}[i] * M'^{mm}_{b}[i])]$$

Where:

a₂₂: is an incorrect prediction in both the *MCKSN* model $M'^{mm}_{b}[i]$ and the human results $H'^{mm}_{b}[i]$. and ***N***: is the total number of all the inferred Cognitive Skill Dependencies(*CSD*). By applying the formula in 6.23, the ME for each *CSD* level is as follows:

For Cognitive Skill Dependency (Understanding), *ME* was equal to 0.0356290588537. By turning it to a percentage, the margin of error was 3.58%, with a confidence level of 95%. This means that there is a 95% chance that the *MCKSN* result for Cognitive Skill Dependency (Understanding), did NOT happen by accident. It can also mean that there is a probability of 3.5% that the Cognitive Skill Dependency (Understanding), using the *MCKSN* model were incorrect.

For Cognitive Skill Dependency (Applying), *ME* was equal to 0.0357876652214. By turning it to a percentage, the margin of error was 3.56%, with a confidence level of 95%. This means that there is a 95% chance that the *MCKSN* result for Cognitive Skill Dependency (Applying) did NOT happen by accident. It can also mean there is a probability of 3.58% that the Cognitive Skill Dependency (Applying) using the *MCKSN* model were misinterpreted.

For Cognitive Skill Dependency (Analyzing-Evaluating), *ME* was equal to 0.037829206789. By turning it to a percentage, the margin of error was 3.78%, with a confidence level of 95%. This means that there is a 95% chance that the *MCKSN* result for

Cognitive Skill Dependency (Analyzing-Evaluating) did NOT happen by accident. It can also mean there is a probability of 3.78% that the Cognitive Skill Dependency (Analyzing-Evaluating) using the *MCKSN* model were misinterpreted.

For Cognitive Skill Dependency (Creating), *ME* was equal to 0.0433301771194. By turning it to a percentage, the margin of error was 4.33%, with a confidence level of 95%. This means that there is a 95% chance that the *MCKSN* result for Cognitive Skill Dependency (Creating) did NOT happen by accident. It can also mean there is a probability of 4.33% that the Cognitive Skill Dependency (Creating)using the *MCKSN* model were misinterpreted.

Table 6.14. Margin of Error and Confidence Level for CSD's .

Cognitive Skill Dependencies	Margin of Errors	Confidence Level
Understanding	3.56%	95%
Applying	3.58%	95%
Analyzing-Evaluating	3.78%	95%
Creating	4.33%	95%

6.8 Summary and Discussion

In this chapter, we came full circle by presenting the results of the proposed model. The material introduced gradually in the previous chapters was here summarized in a meaningful way, producing a clear picture of the critical experiments of this dissertation.

The obtained results of the model were presented in detail. In order to measure the reliability of the proposed model the results were compared with human evaluation using a survey. Step by step explanation of the human subject procedure was presented.

Additionally, an evaluation measures that were used to evaluate the efficiency of the presented model. The model measured the quantitative inference and probability estimation of the inferred Cognitive Skill Dependencies facts. Finally, the analysis of the experiment was displayed.

CHAPTER 7: CONCLUSION AND FUTURE WORK

7.1 Conclusion

In computer sciences curriculum instructors usually focuses on which concepts to teach and when, not on how. An interesting angle to rank the learning materials by level of difficulty in terms of inferring the Cognitive Skill Dependencies. This chapter recapitulate what has come out of our work. The dissertation developed a novel meta learning recommended model to classify the domain specific concepts based on their Cognitive Skill Dependencies. The engine of the model is the third phase, which is inferring Cognitive Skill Dependencies in the learning region. The problem of inferring Cognitive Skill Dependencies was explored, and a sophisticated technique was implemented to access high accuracy and efficiency. Additionally, many of the sub tasks have been solved to reach the optimal result of inferring Cognitive Skill Dependencies.

We recommended to use the initial version of our model to introduce varity of options starting from measuring the knowledge in a textbook, which is affect both the quality of the knowledge acquired and the time needed to learn this knowledge. Furthermore, using *FOL* as a fundamental to construct the *SIR*, which are used for inferring Cognitive Skill Dependencies using *MCKSN* model. Based on the results and the human evaluation our model introduced accurate result where the topic is still open research. Some interesting questions will be addressed as a future work.

The result of this dissertation added a novel parameter, which is using Cognitive Skill Dependencies to improve knowledge quality for the learner's by maximizing the learning benefits with minimum efforts for the learner.

7.2 *MCKSN* Model Application

The model will serve the following application:

- Extracting teaching plan for instructors based on Cognitive Skill Dependencies between concepts.
- Inferring the learning objectives of a course based on Cognitive Skill Dependencies.
- Introducing a learning map for CS learners.
- Generating a summary template from the learning materials based on the most discussion topics in the textbook.

7.3 Limitations of the *MCKSN* Model

Here the limitations of the current model are presented. These limitations are not fundamental to context sensitive moralization; The extensions addressed as future work

- The experiment setup should be created by domain experts (a large group of professors). The access for the domain experts was not available in the current work.
- The participants in this experiment had various grade averages (*A, B, C, and D*). In other words, all grade levels were included b in this experiment

because of the IRB limitation of the sample size used in the human model. Also, if the sample size were smaller, then the possibility that participants could more easily be identified would also increase.

- In these experiments simple Skill Inference Rules (*SIR*) were used. The *SIR* set can extend to include more complex patterns.
- The automatic model needed to be presented as an online tool in website to serve the learner and instructors.
- More learning sources needed to be tested to see the behavior of our model.
- The domain space only CS domain space hopefully other domains be tested as well.
- Only English Language is used in the model.

7.4 Future work

There are several interesting and promising directions in which this work could be extended.

- Do a human experiment which is included only students with high level grade with specific *GPA* level.
- Do an experiment created by a large expert in Algorithm area.
- More complex Skill Inference Rules (*SIR*) should be generated to be tested by the proposed model.
- Using the proposed model to build an online Computer Science Courses.

- Using the proposed model to create exam questions based on the Cognitive Skill Dependencies levels.
- Using the model to investigate different domain for example analyzing health-records or analyzing social network.
- Even though there exist many applications in which a Singular Value Decomposition (*SVD*) is useful, there are a few drawbacks to using the *SVD*. For example, the choice for the number of dimensions *k* to use can be a crucial aspect. In the dissertation, two dimensions were used. Using too many dimensions will add unnecessary noise to the result. The dimensionality reduction needs deeper analysis.
- It is interesting if it were possible to modify the singular value decomposition *SVD.*
- The model could be used to mimic human learning using psychomotor Bloom domain by building a reliable picture of a student's relevant cognitive states during learning.

APPENDIX A

This appendix shows the experiment result for *MCKSN* for the inferred Cognitive Skill Dependencies with different threshold β_m values.

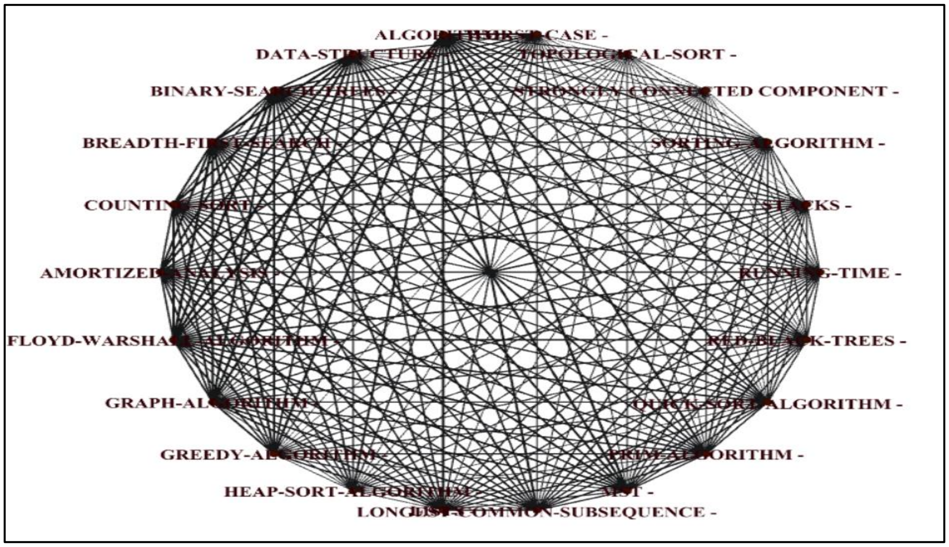


Figure A. 1. a Biredview of the inferred Cognitive Skill Dependencies (Understanding) using MCKSN probability Model at threshold β_m =50%.

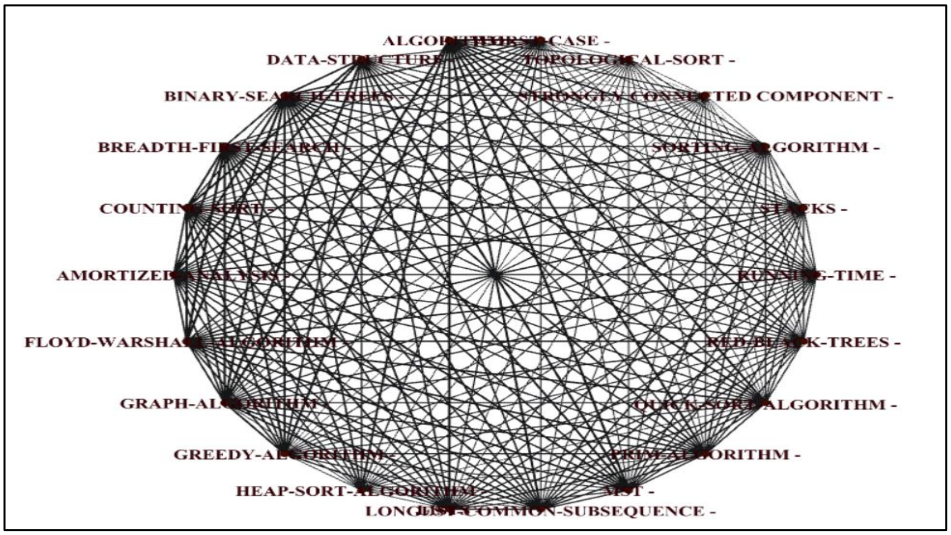


Figure A. 2. a Biredview of the inferred Cognitive Skill Dependencies (Understanding) using MCKSN probability Model at threshold β_m =65%.

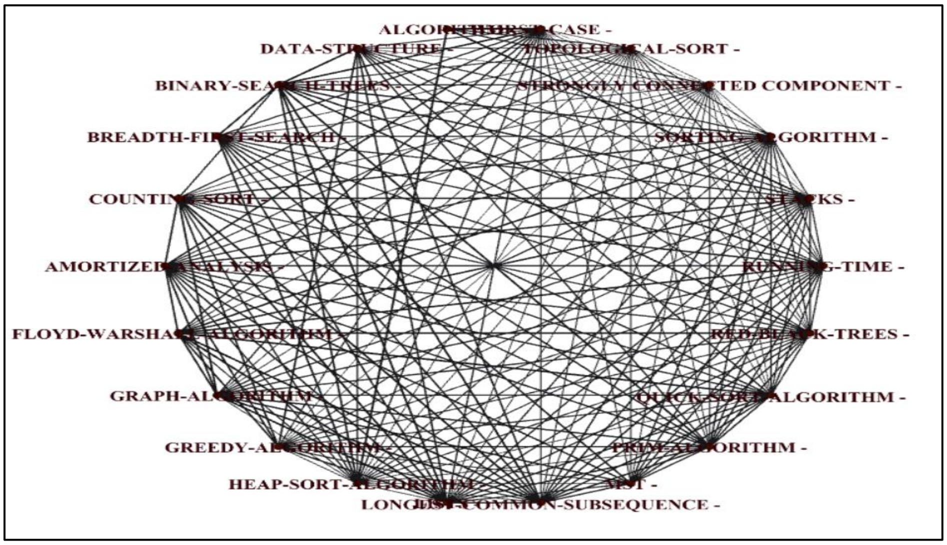


Figure A. 3. a Biredview of the inferred Cognitive Skill Dependencies (Understanding) using MCKSN probability Model at threshold **β_m** =75%.

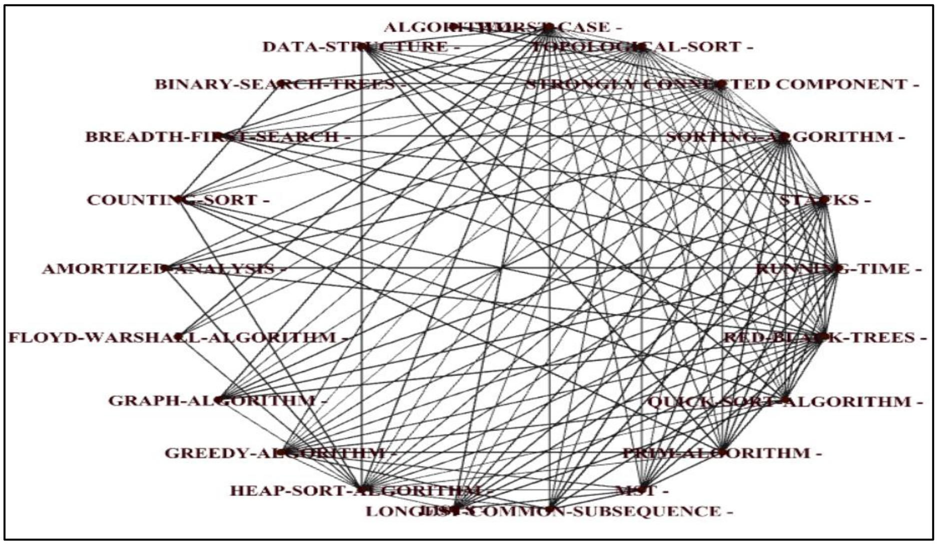


Figure A. 4. a Biredview of the inferred Cognitive Skill Dependencies (Understanding) using MCKSN probability Model at threshold $\boldsymbol{\beta}_{\mathbf{m}}$ =85%.

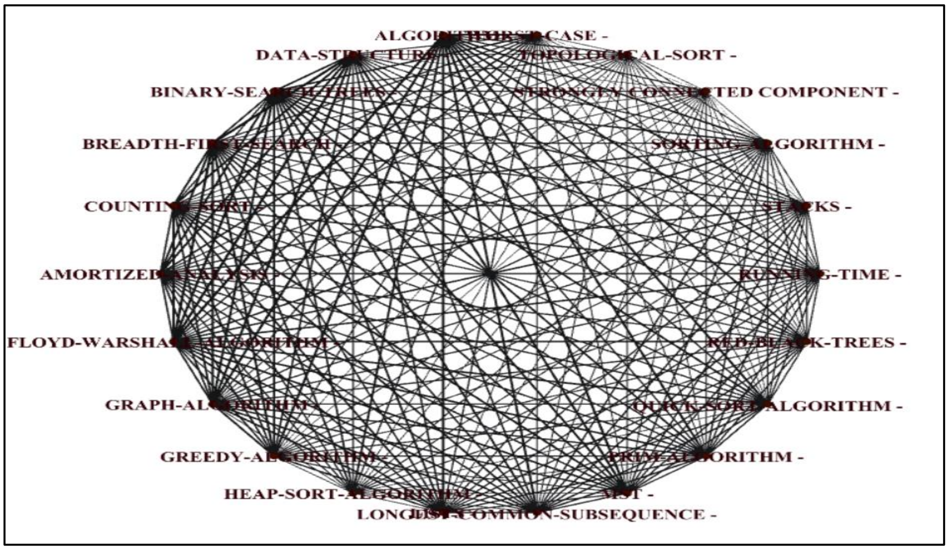


Figure A. 5. a Biredview of the inferred Cognitive Skill Dependencies (Analyzing) using MCKSN probability Model at threshold $\boldsymbol{\beta}_{\mathbf{m}}$ =50%.

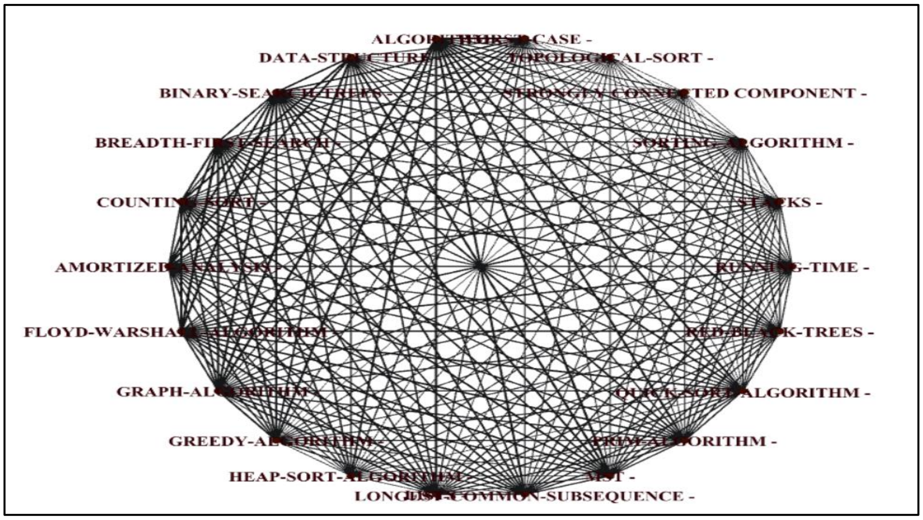


Figure A. 6. a Biredview of the inferred Cognitive Skill Dependencies (Analyzing) using MCKSN probability Model at threshold $\boldsymbol{\beta}_{\mathbf{m}}$ =65%.

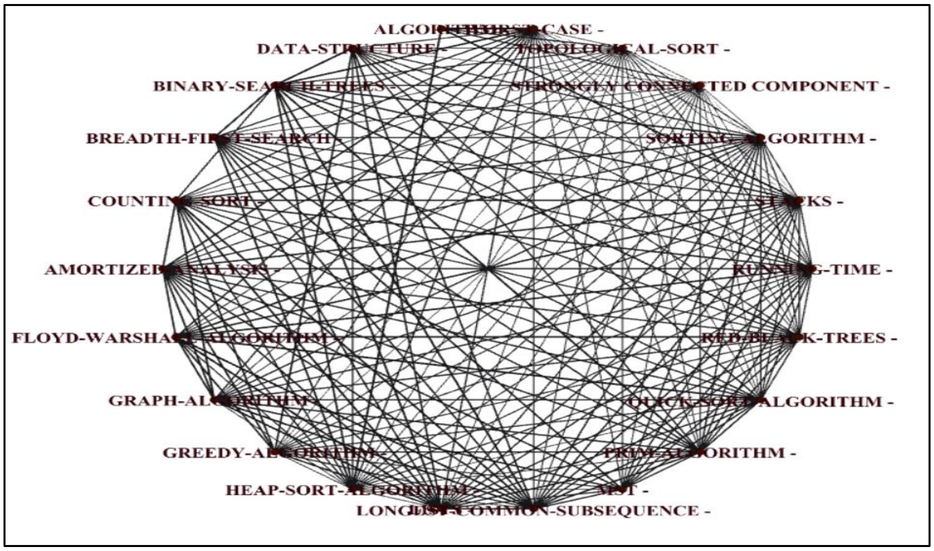


Figure A. 7. a Biredview of the inferred Cognitive Skill Dependencies (Analyzing) using MCKSN probability Model at threshold **βm** =75%.

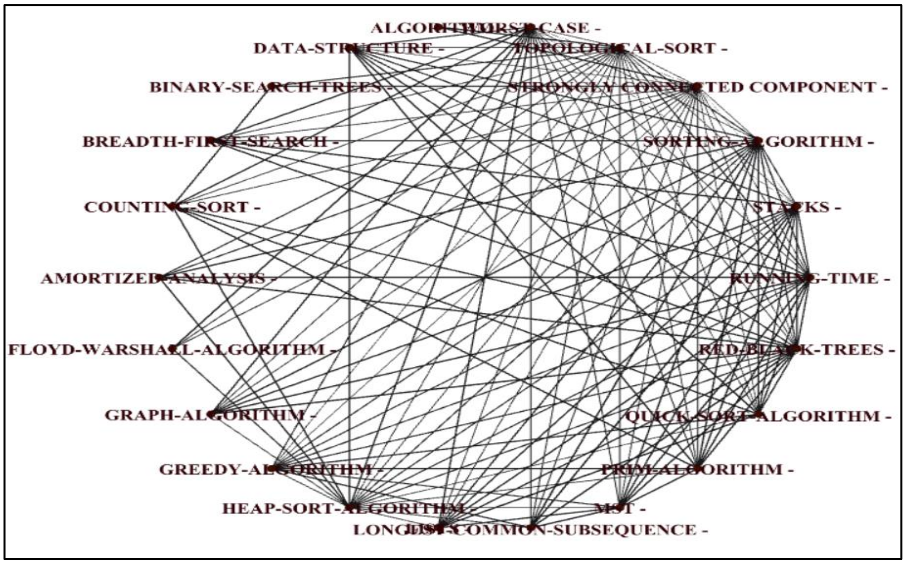


Figure A. 8. a Biredview of the inferred Cognitive Skill Dependencies (Analyzing) using MCKSN probability Model at threshold **βm** =85%.

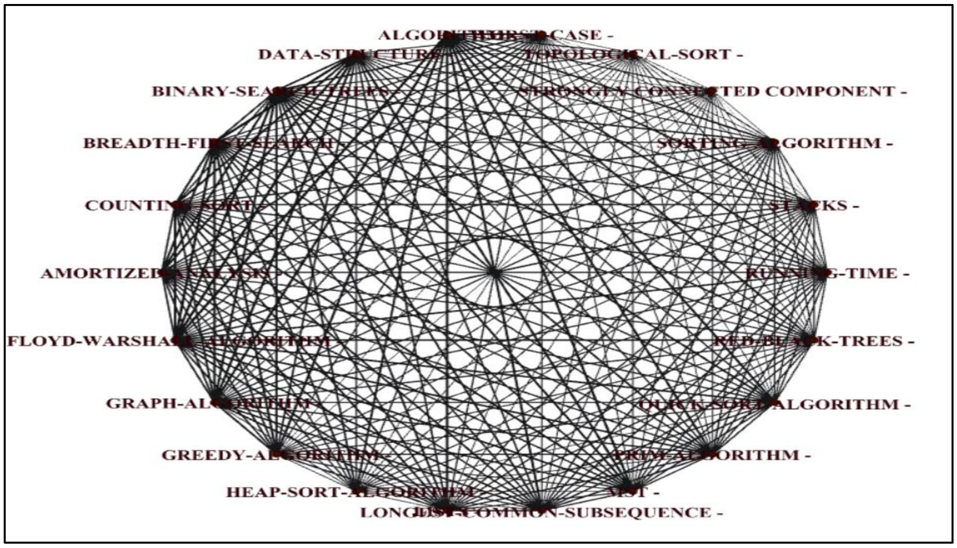


Figure A. 9. a Biredview of the inferred Cognitive Skill Dependencies (Applying) using MCKSN probability Model at threshold **β_m** =50%.

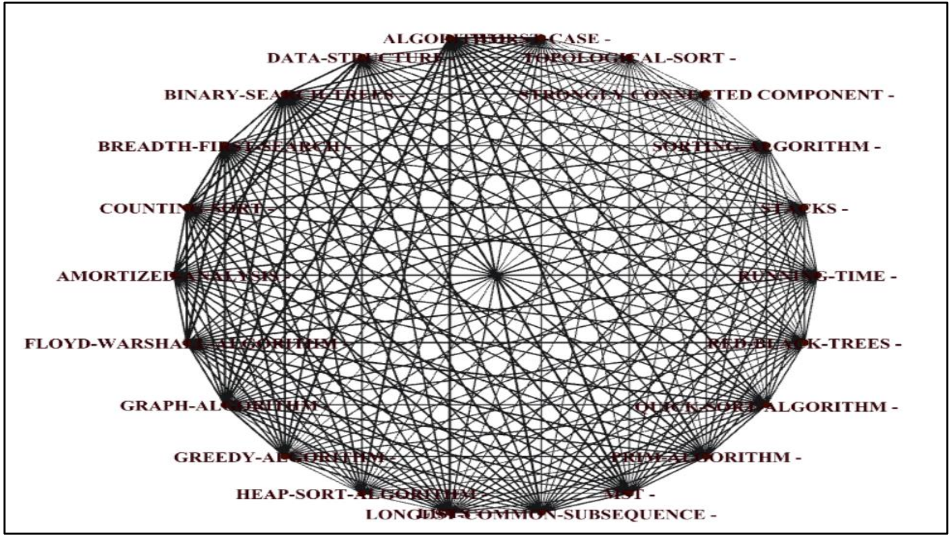


Figure A. 10. a Biredview of the inferred Cognitive Skill Dependencies (Applying) using MCKSN probability Model at threshold **β_m** =65%.

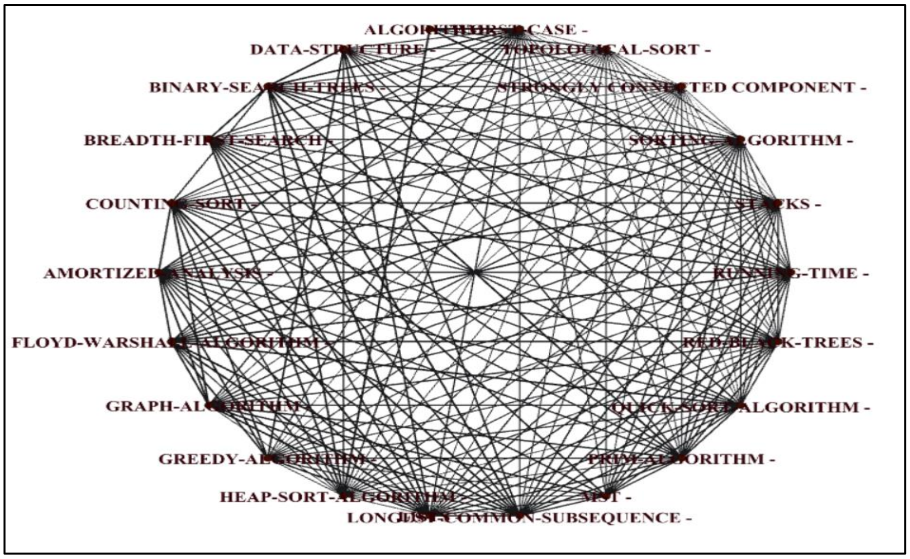


Figure A. 11. a Biredview of the inferred Cognitive Skill Dependencies (Applying) using MCKSN probability Model at threshold **β_m** =75%.

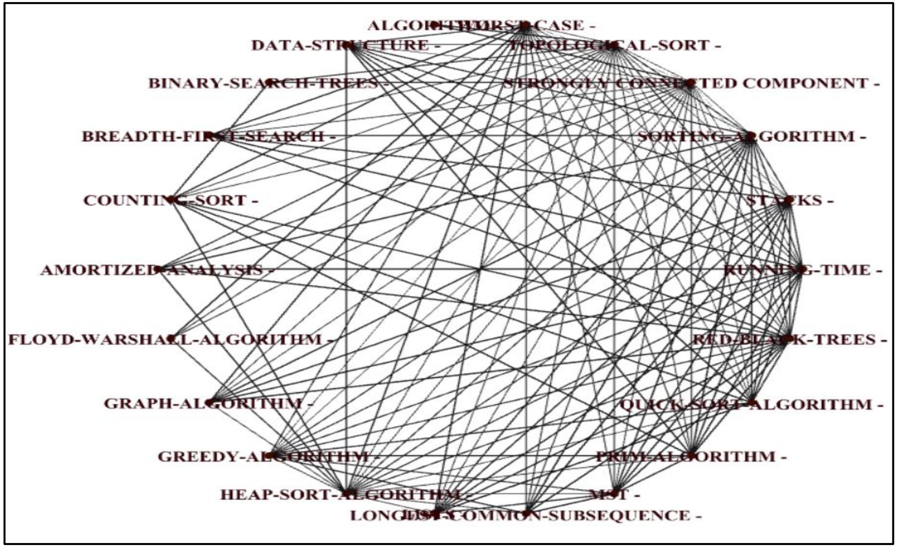


Figure A. 12. a Biredview of the inferred Cognitive Skill Dependencies (Applying) using MCKSN probability Model at threshold **β_m** =85%.

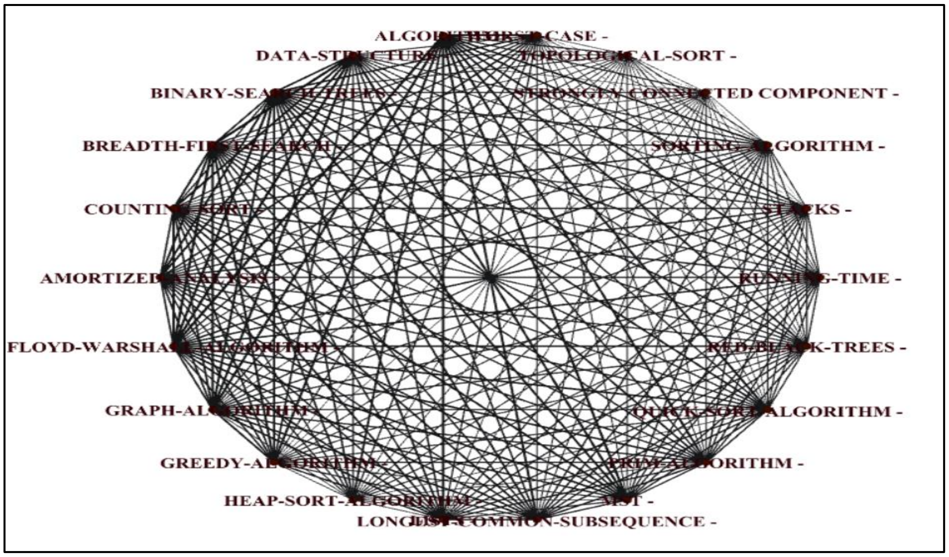


Figure A. 13. a Biredview of the inferred Cognitive Skill Dependencies (Creating) using MCKSN probability Model at threshold **β_m** =50%.

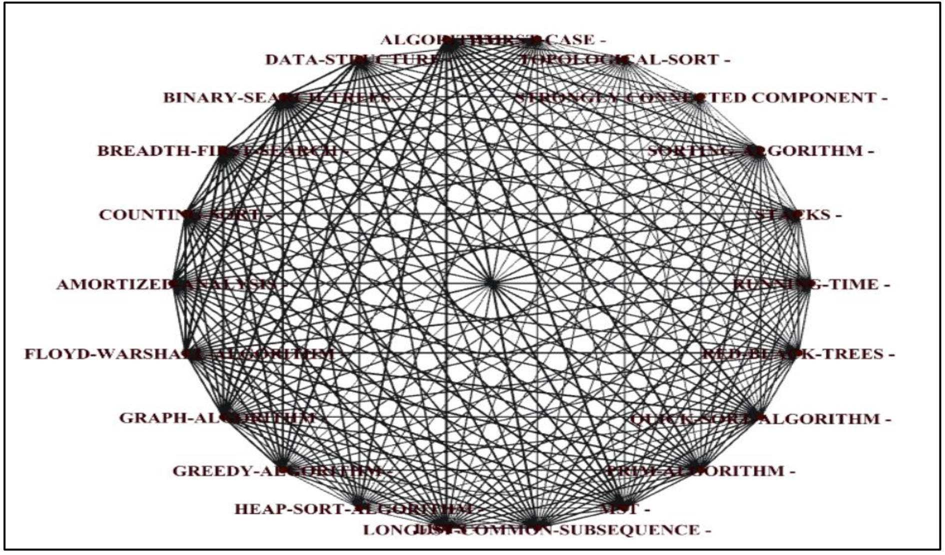


Figure A. 14. a Biredview of the inferred Cognitive Skill Dependencies (Creating) using MCKSN probability Model at threshold **β_m** =65%.

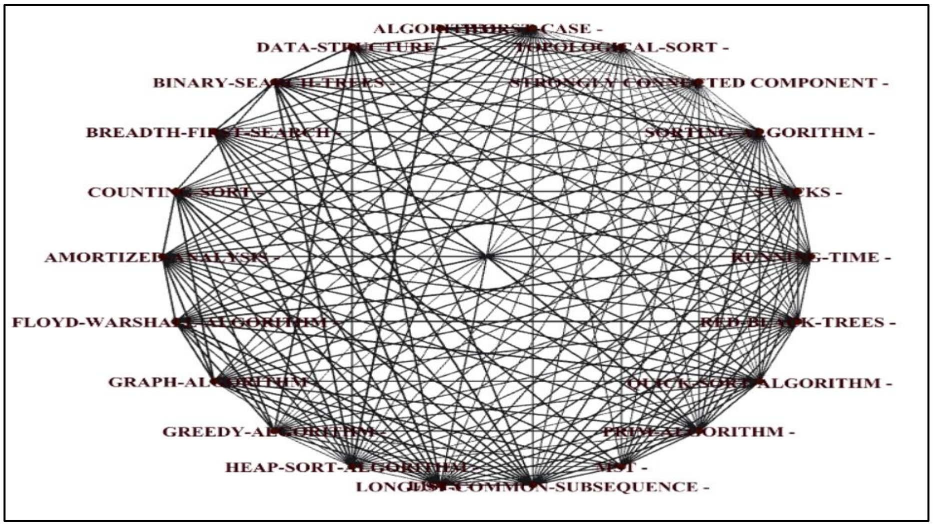


Figure A. 15. . a Biredview of the inferred Cognitive Skill Dependencies (Creating) using MCKSN probability Model at threshold **β_m** =75%.

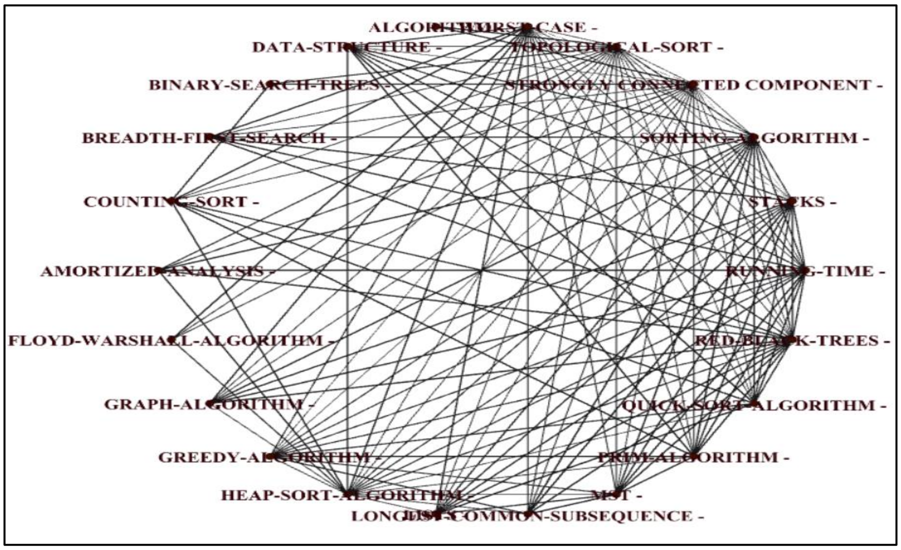


Figure A. 16. a Biredview of the inferred Cognitive Skill Dependencies (Creating) using MCKSN probability Model at threshold **β_m** =85%.

APPENDIX B

AUTHORIZATION OF STUDY BY INSTITUTIONAL REVIEW BOARD

Date: This application was approved on **October 31, 2016.**

Subject: IRB Study Approved

The Kent State University Institutional Review Board has reviewed and approved your Application for Approval to Use Human Research Participants as Level I/Exempt from Annual review research. Your research project involves minimal risk to human subjects and meets the criteria for the following category of exemption under federal regulations.

APPENDIX C

This appendix shows the Cognitive Class for some of the CS-Verbs.

CS-Verb	Cognitive Class
Demonstrate	BL_1, BL_2
Analyze	BL_1, BL_2
Show	BL_1, BL_2
Translate	BL_1, BL_2
Identify	BL_1, BL_2, BL_3
Illustrate	BL_1, BL_2, BL_3
Select	BL_1, BL_2, BL_3, BL_4
Develop	BL_1, BL_2, BL_4
Characterize	BL_1, BL_3
List	BL_1, BL_3
Compare	BL_1, BL_3, BL_4
Estimate	BL_1, BL_3, BL_4
Interpret	BL_1, BL_3, BL_4
Discuss	BL_1, BL_4
Summarize	BL_1, BL_4
Determinant	BL_2, BL_3
Discover	BL_2, BL_3
Examine	BL_2, BL_3
Investigate	BL_2, BL_3
Choose	BL_2, BL_3, BL_4
Relate	BL_2, BL_3, BL_4
Build	BL_2, BL_4
Change	BL_2, BL_4
Construct	BL_2, BL_4
Organize	BL_2, BL_4
Produce	BL_2, BL_4
Solve	BL_2, BL_4
Evaluate	BL_3, BL_4
Measure	BL_3, BL_4

APPENDIX D

13 research papers have been published on the above discussed contributions, where 8 of them are specific to the proposed model in this dissertation. The conferences are peer reviewed, and the papers are as follows:

1. Conceptualize the Domain Knowledge Space in the Light of Cognitive Skills. **F. Nafa**, J.I Khan. Proceedings of the 7th International Conference on Computer Supported Education (CSEDU) **2015**.
2. An Iterative Method for Enhancing Text Comprehension by Automatic Reading of References. Babour, **F. Nafa**, and J. I. Khan. Fourth International Conference on Intelligent Systems and Applications (INTELLI), **2015**
3. Connecting the Dots in a Concept Space by Iterative Reading of FreeText References with Wordnet. Babour, **F. Nafa**, and J. I. Khan. IEEE/WIC/ACM International Conference on Web Intelligence (WI) **2015**
4. Automatic Concepts' Classification Based on Bloom's Taxonomy Using Text Analysis and the Naïve Bayes Classifier Method. **F. Nafa**., S. Othman, and J.I. Khan. International Conference on Computer Supported Education (CSEDU) **2016**
5. Mining Cognitive Skills Levels of Knowledge Units in Text Using Graph Triangularity Mining. **F. Nafa**, S. Othman, J.I. Khan, and A. Babour. IEEE/WIC/ACM International Conference on Web Intelligence (WI) **2016**
6. Discovering Bloom Taxonomic Relationships between Knowledge Units Using Semantic Graph Triangularity Mining. **F. Nafa**, J.I. Khan, S. Othman, and A. Babour. International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC) **2016**
7. Semantic Graph Transitivity for Discovering Bloom Taxonomic Relationships between Knowledge Units in a Text. **F. Nafa**, S. Othman, J.I., Khan, and A. Babour. Proceedings

of the Fifth International Conference on Intelligent Systems and Applications (INTELLI) **2016**

8. Extending Cognitive Skill Classification of Common Verbs in the Domain of Computer Science Algorithms Knowledge Units. **F. Nafa**, S. Othman, and J.I. Khan. Proceedings of the 9th International Conference on Computer Supported Education (CSEDU) **2017**.

REFERENCES

Adorni, G., & Zock, M. (1996). Trends in Natural Language Generation-An Artificial Intelligence Perspective: Fourth European Workshop, EWNLG'93, Pisa, Italy, April 28-30, 1993 Selected Papers (Vol. 1036): Springer Science & Business Media.

Anderson, L. W., Krathwohl, D. R., Airasian, P. W., Cruikshank, K. A., Mayer, R. E., Pintrich, P. R., . . . Wittrock, M. C. (2001). A taxonomy for learning, teaching, and assessing: A revision of Bloom’s taxonomy of educational objectives, abridged edition. *White Plains, NY: Longman*.

Ankan, A., & Panda, A. (2015). *pgmpy: Probabilistic graphical models using python.* Paper presented at the Proceedings of the 14th Python in Science Conference (SCIPY 2015).

Baker, C. F., Fillmore, C. J., & Lowe, J. B. (1998). *The berkeley framenet project.* Paper presented at the Proceedings of the 17th international conference on Computational linguistics-Volume 1.

Biegler, L. T. (2010). *Nonlinear programming: concepts, algorithms, and applications to chemical processes* (Vol. 10): Siam.

Bloom, B. S. (1956). Taxonomy of educational objectives. Vol. 1: Cognitive domain. *New York: McKay*, 20-24.

Bloom, B. S., Engelhart, M. D., Furst, E. J., Hill, W. H., & Krathwohl, D. R. (1956). *Taxonomy of educational objetives: the classification of educational goals: handbook I: cognitive domain*. Retrieved from

BLOOM’S, T. M. E. (1965). *Bloom’s taxonomy of educational objectives*: Longman.

Bourque, P., Buglione, L., Abran, A., & April, A. (2003). *Bloom? s Taxonomy Levels for Three Software Engineer Profiles.* Paper presented at the null.

Brownlee, K. A. (1965). *Statistical theory and methodology in science and engineering* (Vol. 150): Wiley New York.

Buck, D., & Stucki, D. J. (2001). JKarelRobot: a case study in supporting levels of cognitive development in the computer science curriculum. *ACM SIGCSE Bulletin, 33*(1), 16-20.

Casella, G., & George, E. I. (1992). Explaining the Gibbs sampler. *The American Statistician, 46*(3), 167-174.

Cheng, G., Wan, Y., Buckles, B. P., & Huang, Y. (2014). *An introduction to Markov logic networks and application in video activity analysis.* Paper presented at the Computing, Communication and Networking Technologies (ICCCNT), 2014 International Conference on.

Clifford, P. (1990). Markov random fields in statistics. *Disorder in physical systems: A volume in honour of John M. Hammersley, 19*.

Crossland, Z. (2010). Materiality and embodiment *The Oxford Handbook of Material Culture Studies*.

Crowe, A., Dirks, C., & Wenderoth, M. P. (2008). Biology in bloom: implementing Bloom's taxonomy to enhance student learning in biology. *CBE—Life Sciences Education, 7*(4), 368-381.

Dang, H. T. (2004). Investigations into the role of lexical semantics in word sense disambiguation.

Davis, J., & Goadrich, M. (2006). *The relationship between Precision-Recall and ROC curves.* Paper presented at the Proceedings of the 23rd international conference on Machine learning.

de Oliveira, P. C. (2009). Probabilistic reasoning in the semantic web using markov logic. *Master's Thesis, University of Coimbra*.

Doran, M. V., & Langan, D. D. (1995). *A cognitive-based approach to introductory computer science courses: lesson learned.* Paper presented at the ACM SIGCSE Bulletin.

Dowty, D. R. (2012). *Word meaning and Montague grammar: The semantics of verbs and times in generative semantics and in Montague's PTQ* (Vol. 7): Springer Science & Business Media.

Fawcett, T. (2006). An introduction to ROC analysis. *Pattern recognition letters, 27*(8), 861-874.

Fitting, M. (2012). *First-order logic and automated theorem proving*: Springer Science & Business Media.

Foltz, P. W., Kintsch, W., & Landauer, T. K. (1998). The measurement of textual coherence with latent semantic analysis. *Discourse processes, 25*(2-3), 285-307.

Friedman, N. (2004). Inferring cellular networks using probabilistic graphical models. *Science, 303*(5659), 799-805.

Gad-Elrab, M. H., Stepanova, D., Urbani, J., & Weikum, G. (2016). *Exception-enriched rule learning from knowledge graphs.* Paper presented at the International Semantic Web Conference.

Geman, S., & Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*(6), 721-741.

Getoor, L., & Taskar, B. (2007). *Introduction to statistical relational learning* (Vol. 1): MIT press Cambridge.

Gilbert, R. O. (1987). *Statistical methods for environmental pollution monitoring*: John Wiley & Sons.

Golub, G. H., & Reinsch, C. (1970). Singular value decomposition and least squares solutions. *Numerische mathematik, 14*(5), 403-420.

Grishman, R., & Sundheim, B. (1996). *Message understanding conference-6: A brief history*. Paper presented at the COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics.

Guo, S., Wang, Q., Wang, L., Wang, B., & Guo, L. (2016). *Jointly embedding knowledge graphs and logical rules*. Paper presented at the Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing.

Hasdorff, L. (1976). Gradient optimization and nonlinear control.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications.

Hernán-Losada, I., Pareja-Flores, C., & Velázquez-Iturbide, J. Á. (2008). *Testing-Based Automatic Grading: a proposal from Bloom's taxonomy*. Paper presented at the Advanced Learning Technologies, 2008. ICALT'08. Eighth IEEE International Conference on.

Ivanova, O. (2017). Applied Concepts of Probabilistic Programming. *Prozesse, Technologie, Anwendungen, Systeme und Management*, 161.

Jahn, J. (2007). *Introduction to the theory of nonlinear optimization*: Springer Science & Business Media.

Jiang, S., Pang, G., Wu, M., & Kuang, L. (2012). An improved K-nearest-neighbor algorithm for text categorization. *Expert Systems with Applications, 39*(1), 1503-1509.

Johnson, C. G., & Fuller, U. (2006). *Is Bloom's taxonomy appropriate for computer science?* Paper presented at the Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006.

Kemeny, J. G., Snell, J. L., & Knapp, A. W. (2012). *Denumerable Markov chains: with a chapter of Markov random fields by David Griffeath* (Vol. 40): Springer Science & Business Media.

Khairuddin, N. N., & Hashim, K. (2008). *Application of Bloom's taxonomy in software engineering assessments.* Paper presented at the Proceedings of the 8th WSEAS International Conference on Applied Computer Science.

Khan, J., & Hardas, M. (2007). A technique for representing course knowledge using ontologies and assessing test problems *Advances in Intelligent Web Mastering* (pp. 174-179): Springer.

Kipper, K., Dang, H. T., & Palmer, M. (2000). Class-based construction of a verb lexicon. *AAAI/IAAI, 691*, 696.

Kipper, K., Korhonen, A., Ryant, N., & Palmer, M. (2006). *Extending VerbNet with novel verb classes.* Paper presented at the Proceedings of LREC.

Klavans, J., & Kan, M.-Y. (1998). *Role of verbs in document analysis.* Paper presented at the Proceedings of the 17th international conference on Computational linguistics-Volume 1.

Kolb, D. A. (2014). *Experiential learning: Experience as the source of learning and development*: FT press.

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*: MIT press.

Lewis, D. D. (1995). *Evaluating and optimizing autonomous text classification systems.* Paper presented at the Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval.

Li, S. Z. (2009). *Markov random field modeling in image analysis*: Springer Science & Business Media.

Lister, R. (2000). *On blooming first year programming, and its blooming assessment.* Paper presented at the Proceedings of the Australasian conference on Computing education.

Lister, R., & Leaney, J. (2003). Introductory programming, criterion-referencing, and bloom. *ACM SIGCSE Bulletin, 35*(1), 143-147.

Lowd, D., & Domingos, P. (2007). *Efficient weight learning for Markov logic networks.* Paper presented at the European Conference on Principles of Data Mining and Knowledge Discovery.

Lyons, J. (1968). *Introduction to theoretical linguistics*: Cambridge university press.

Lyons, J. (1995). *Linguistic semantics: An introduction*: Cambridge University Press.

Machanick, P. (2000). *Experience of applying Bloom's Taxonomy in three courses.* Paper presented at the Proc. Southern African Computer Lecturers' Association Conference.

Makkai, M., & Reyes, G. E. (2006). *First order categorical logic: model-theoretical methods in the theory of topoi and related categories* (Vol. 611): Springer.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., & McClosky, D. (2014). *The Stanford CoreNLP natural language processing toolkit.* Paper presented at the Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations.

Manning, C. D., Manning, C. D., & Schütze, H. (1999). *Foundations of statistical natural language processing*: MIT press.

Margaritis, D., & Thrun, S. (2000). *Bayesian network induction via local neighborhoods.* Paper presented at the Advances in neural information processing systems.

Mario&Matr. (2014). Introduction to Markov Random Fields and Markov Logic Networks.

McHugh, M. L. (2012). Interrater reliability: the kappa statistic. *Biochemia medica: Biochemia medica, 22*(3), 276-282.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *J. Chem. Phys, 21*(6), 1087-1092.

Miller, G. (1998). *WordNet: An electronic lexical database*: MIT press.

Miller, K. J. (1998). Modifiers in wordnet. *WordNet: an electronic lexical database*, 47-67.

Mishra, M., Huan, J., Bleik, S., & Song, M. (2012). *Biomedical text categorization with concept graph representations using a controlled vocabulary.* Paper presented at the Proceedings of the 11th International Workshop on Data Mining in Bioinformatics.

Mitra, P., Wiederhold, G., & Kersten, M. (2000). *A graph-oriented model for articulation of ontology interdependencies.* Paper presented at the International Conference on Extending Database Technology.

Muller, A. C., & Guido, S. (2017). *Introduction to machine learning with Python: a guide for data scientists*: O'Reilly Media.

Müller, A. C., & Guido, S. (2016). *Introduction to machine learning with Python: a guide for data scientists*: " O'Reilly Media, Inc.".

Nafa, F., & Khan, J. (2015). *Conceptualize the Domain Knowledge Space in the Light of Cognitive Skills.* Paper presented at the Proceedings of the 7th International Conference on Computer Supported Education-Volume 1.

Nafa, F., Khan, J. I., & Othman, S. (2017). *Extending Cognitive Skill Classification of Common Verbs in the Domain of Computer Science for Algorithms Knowledge Units.* Paper presented at the CSEDU (1).

Nafa, F., Khan, J. I., Othman, S., & Babour, A. (2016a). *Discovering Bloom Taxonomic Relationships between Knowledge Units Using Semantic Graph Triangularity Mining.* Paper presented at the 2016 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC).

Nafa, F., Khan, J. I., Othman, S., & Babour, A. (2016b). *Mining Cognitive Skills Levels of Knowledge Units in Text Using Graph Tringluarity Mining.* Paper presented at the Web Intelligence Workshops (WIW), IEEE/WIC/ACM International Conference on.

Nafa, F., Khan, J. I., Othman, S., & Babour, A. (2016c). Semantic Graph Transitivity for Discovering Bloom Taxonomic Relationships between Knowledge Units in a Text. *INTELLI 2016*, 134.

Nevid, J. S., & McClelland, N. (2013). Using Action Verbs as Learning Outcomes: Applying Bloom's Taxonomy in Measuring Instructional Objectives in Introductory Psychology. *Journal of Education and Training Studies, 1*(2), 19-24.

Oliver, D., & Dobele, T. (2007). First year courses in IT: A bloom rating. *Journal of Information Technology Education: Research, 6*, 347-360.

Parham, J., Chinn, D., & Stevenson, D. (2009). *Using Bloom's taxonomy to code verbal protocols of students solving a data structure problem.* Paper presented at the Proceedings of the 47th Annual Southeast Regional Conference.

Pearl, J. (2014). *Probabilistic reasoning in intelligent systems: networks of plausible inference*: Elsevier.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., . . . Dubourg, V. (2011). Scikit-learn: Machine learning in Python. *Journal of machine learning research, 12*(Oct), 2825-2830.

Powers, D. M. (2011). Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation.

Reigeluth, C. M. (2013). *Instructional design theories and models: An overview of their current status*: Routledge.

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine learning, 62*(1-2), 107-136.

Robert, C. (2014). Machine learning, a probabilistic perspective: Taylor & Francis.

Russell, S. J., & Norvig, P. (2016). *Artificial intelligence: a modern approach*: Malaysia; Pearson Education Limited.

Schuler, K. K. (2005). VerbNet: A broad-coverage, comprehensive verb lexicon.

Schulte, C., & Bennedsen, J. (2006). *What do teachers teach in introductory programming?* Paper presented at the Proceedings of the second international workshop on Computing education research.

Scott, T. (2003). Bloom's taxonomy applied to testing in computer science classes. *Journal of Computing Sciences in Colleges, 19*(1), 267-274.

Shi, L., & Mihalcea, R. (2005). *Putting pieces together: Combining FrameNet, VerbNet and WordNet for robust semantic parsing.* Paper presented at the International conference on intelligent text processing and computational linguistics.

Smullyan, R. R. (2012). *First-order logic* (Vol. 43): Springer Science & Business Media.

Sowa, J. F. (2000). *Knowledge representation: logical, philosophical, and computational foundations* (Vol. 13): Brooks/Cole Pacific Grove, CA.

Starr, C. W., Manaris, B., & Stalvey, R. H. (2008). *Bloom's taxonomy revisited: specifying assessable learning objectives in computer science.* Paper presented at the ACM SIGCSE Bulletin.

Sutton, C., & McCallum, A. (2006). *An introduction to conditional random fields for relational learning* (Vol. 2): Introduction to statistical relational learning. MIT Press.

Swier, R. S., & Stevenson, S. (2004). *Unsupervised Semantic Role Labellin.* Paper presented at the Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing.

Thompson, E., Luxton-Reilly, A., Whalley, J. L., Hu, M., & Robbins, P. (2008). *Bloom's taxonomy for CS assessment.* Paper presented at the Proceedings of the tenth conference on Australasian computing education-Volume 78.

Thompson, T. (2008). Mathematics teachers' interpretation of higher-order thinking in Bloom's taxonomy. *International electronic journal of mathematics education, 3*(2), 96-109.

Tur, G., Hakkani-Tür, D., & Schapire, R. E. (2005). Combining active and semi-supervised learning for spoken language understanding. *Speech Communication, 45*(2), 171-186.

Urbanek, S., & Theus, M. (2008). *Interactive graphics for data analysis: principles and examples*: Chapman and Hall/CRC.

Valle, K., & Ozturk, P. (2011). *Graph-based representations for text classification.* Paper presented at the India-Norway Workshop on Web Concepts and Technologies, Trondheim, Norway.

Van Rijsbergen, C. J. (1974). Foundation of evaluation. *Journal of Documentation, 30*(4), 365-373.

Wall, M. E., Rechtsteiner, A., & Rocha, L. M. (2003). Singular value decomposition and principal component analysis *A practical approach to microarray data analysis* (pp. 91-109): Springer.

Walter, C. (2004). Transfer of reading comprehension skills to L2 is linked to mental representations of text and to L2 working memory. *Applied Linguistics, 25*(3), 315-339.

Wasserman, S., & Pattison, P. (1996). Logit models and logistic regressions for social networks: I. An introduction to Markov graphs andp. *Psychometrika, 61*(3), 401-425.

Winkler, G. (2012). *Image analysis, random fields and Markov chain Monte Carlo methods: a mathematical introduction* (Vol. 27): Springer Science & Business Media.