Semantic Graph Transitivity for Discovering Bloom Taxonomic Relationships between Knowledge Units in a Text

Fatema Nafa, Javed I. Khan, Salem Othman, and Amal Babour
Department of Computer Science, Kent State University
Kent, Ohio, USA
Email : fnafa, Javed, sothman, ababour@kent.edu

***Abstract*— Manual inferring of semantic relationships by domain experts is an expensive and time consuming task; thus, automatic techniques are needed. In this paper, we propose an automatic novel technique for inferring cognitive relationships among concepts and knowledge units in the learning resources by using Graph-transitivity. The cognitive relationships are expressed as Bloom Taxonomy levels. Learning resources are represented as knowledge units in texts. The technique determines significant relationships among knowledge units by utilizing transitivity of knowledge units in the computer science domain. We share an experiment that evaluates and validates the technique from three textbooks. The performance analysis shows that the technique succeeds in discovering the hidden cognitive relationships among knowledge units in learning resources.**

Keywords*— *Cognitive Graph; Graph Transitivity; Knowledge Unit; Graph Mining; Bloom Taxonomy.

I. INTRODUCTION

Extracting semantic relationships from a text has been widely studied in several research including Natural Language Processing (NLP), Text Mining, Information Retrieval (IR), and others. The goal of the relationships' extraction is different from one task to another and from one resource to another. Learning resources are the most significant repositories of knowledge and information. Discovering hidden interconnections among knowledge units is interesting. Hidden interconnections are represented in different forms. In this paper, the interconnections among knowledge units are represented as a cognitive theory called Bloom Taxonomy(BT). The concept of cognitive theory has crossed the line from psychology and educational theory and has become an important part of computer technology research. The taxonomy idea was first introduced by Benjamin Bloom. Bloom identified three domains of educational activities: the cognitive domain (mental skills), the affective domain (growth in feelings or emotional areas), and the psychomotor domain (physical skills) [1]. The cognitive domain is divided into six levels: 1) knowledge, 2) comprehension, 3) application, 4) analysis, 5) synthesis, and 6) evaluation. The Bloom model was modified in 2001 by Anderson and a team of cognitive psychologists [2]. Significant changes were made to the Bloom's Taxonomy model. The original taxonomy of educational objectives, is referred to as Bloom's Taxonomy and Anderson's work, is known as Revised Bloom's Taxonomy [2]. Revised Bloom's was modified to the Computer Science based Cognitive Domain (CSCD) [3] to make it appropriate for the concept domain in computer science. For this paper, only the cognitive domain is used and we discussed the first sub-task in our previous work [3]. We are going to discuss the second sub-task in this paper.

We introduced an automatic technique to infer the relationships based on CSCD levels among knowledge units using the graph transitivity. The CSCD is used to identify and progressively measure of learner's cognitive level. A learner is not expected to understand the text based on the given ordered knowledge units. Thus, a shared language is needed to provide a highlighted learning map of a text based on cognitive skills.

The rest of the paper is organized as follows. The related work is presented in Section II. The problem definition is discussed in Section III. Section IV describes an overview of the system. Section V describes the transitivity technique as well as a description of the algorithm in detail. Section VI presents the classification of the knowledge units. Section VII shows examples of the technique. The experiment setup and an evaluation of the technique are explained in Section VIII. Section IX presents the conclusion and future work

II. RELATED WORK

The work presented in this paper is situated at the intersection of several areas of related prior work from the linguistics perspective, Graph perspective, and Graph Transitivity Property perspective. We will discuss each of these in turn.

From the *linguistics* perspective, theorists developed 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 [1] is known as Bloom's Taxonomy. Bloom's Taxonomy has been applied in the field of computer science for various purposes such as managing course design [4], measuring the cognitive difficulty levels of computer science materials [4], and structuring assessments [5]. Bloom's Taxonomy has also been used in grading as an alternative to grading on a curve [6]. Additionally, from the mining perspective, there has been some interesting research about extracting relations among concepts. Relations could be replaced by the synonym relationships, or a hypernym, an association, etc. [7] [8]. These relationships are successfully used in different domains and applications [9].

From the *Graph* Perspective, the representation of the extracted relationship is the graph. There has been some research on graphical text representation such as concept graphs [10] and ontology [11]. The authors proposed Concept Graph Learning to present relations among concepts from prerequisite relations among courses.

From the *Graph Transitivity Property* perspective, in the definition of transitivity in graph, two nodes are connected if they share a direct neighbor, so the inferred

 ISBN: 978-1-61208-518-0

hidden relationship between those two nodes is based on the transitivity. There has been research on the transitivity in the domain of Biology [12]. In addition, transitivity has been studied in friendship graphs in social networks research [13].

None of the previous work handles the problem of inferring the relationships among knowledge units based on *CSCD* levels in the domain of computer science. Using graph transitivity is a promising way to reach this goal and discover novel cognitive relationships between knowledge units.

This paper presents a technique for mining *CSCD* levels among the knowledge units in a textbook. The technique is based on using graph transitivity to discover relationships between knowledge units. Transitivity technique describes levels of increasing complexity in students understanding of knowledge units. It has the flexibility of giving the new sequential ordering of the knowledge units in a textbook. According to the experimental evaluations, the method can efficiently identify *CSCD* levels among knowledge units. Building an automatic technique to assist in organizing knowledge units based on the level of cognitive skills will provide a new learning trajectory for the learners and will help in circumventing their deficit in understanding any textbook.

III. PROBLEM DEFINITION

In this section let us introduce some definitions, which are used in this paper.

Concepts: are terms that have significant meaning(s) in computer sciences.

Knowledge Unit (KU): is defined as a group of sentences that discuss specific topics in computer sciences and consist of concepts, which are related to the topic.

The Overlap between Knowledge Unit: if a KU consist group of concepts and at least one of these concepts appears in another KU then, we say the two KU's are overlapping.

Given a textbook T^B that contains a set of knowledge units (it could be a topic from a textbook or more) KU = $\{ku_1, ku_2 \ldots ku_n\}$ where *n* is the number of knowledge units in T^B. Each knowledge unit is a group of sentences $ku_i = \{s_1, s_2 \ldots s_m\}$ and each sentence is composed of a sequence of concepts $s_i = \{c_1,..,c_o\}$; in addition, the Computer Science based Cognitive Domain (*CSCD*) has the levels of (Understanding, Analyzing, Applying-Evaluating, and Creating),which are denoted as $\{B_1, B_2, B_3, B_4\}$ respectively [3]. Each level has a subset of measurable verbs. The contribution of this work is to find transitivity function $f(x)$: KU→βi which maps knowledge units (KU) according to Computer-Science based Cognitive Domain (*CSCD*) using the subset of measurable verbs. To handle this problem, we create a semantic graph G^S from the given textbook T^B in order to find the relationships among concepts in the knowledge unit and subsequently relationships among knowledge units themselves. The transitivity will output the hidden links among knowledge units according to *CSCD* levels. For example, consider that from a textbook three knowledge units (KU #1, KU #2, and KU #3) have been chosen, and we need to find a relationship between knowledge units based on *CSCD* levels using transitivity. As described in the problem definition we have two main problems:

- Converting a textbook to a semantic graph G^S is a directed graph G^S = (C, V) where C (concept) represents nodes, and V (verbs) represents the labels of the relationship among concepts.
- Finding out the relationship (X) among knowledge unit#1, knowledge unit#2 and knowledge unit#3 based on (*CSCD*) βi= $\{B_1, B_2, B_3, B_4\}$. Fig. 1 represents the sub-part of G^S for three knowledge units where some of the relationships mainly exist among the KU from the same textbook, or from a textbook of similar topics.

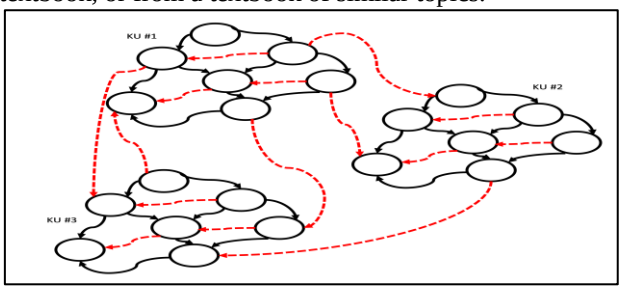


Figure 1. Three Knowledge Units and the Relationships Between them.

IV. OVERVIEW OF THE SYSTEM

The system consists of three main components, called, *System Core*, *CSCD Engine*, and *Domain Lexicon* as in Fig. 2. The input for the system is a textbook and the output is a cognitive graph G^C classified into *CSCD* levels.

The system *Core* was presented in detailed in our previous work [3]. It includes the following four parts: Text-Preprocessing, Natural Language Processing (NLP), Domain Specific Extraction, and Semantic Relationship Extraction. In the Text-Preprocessing part, the system assumes that the input files are in plain text format. Any other formats are turned into a plain text before it starts the other steps. Then, the Natural Language Processing part incorporates NLP tools, such as splitting each sentence into a sequence of tokens where tokens are unique concepts. Stanford Parser, which is used to parse each sentence to get its part-of-speech (verb, noun, adjective, etc.) will be used to extract semantic relationships between concepts. The Domain Specific Extraction part contains concepts related to the domain of interest, which is computer science. We build the specific stop words list manually, because there is no stop list related to the domain under study. It can also be updated during the process of the system. The Semantic Relationship Extraction part includes extracting the relationships in the form of concept-verb-concept, among concepts in the knowledge unit. The final form of extraction is represented as a semantic graph. Lyons and other [15] structural linguists hold that "words cannot be defined independently from other words. A word's relationship with other words is part of the meaning of the word".

CSCD Engine: The CSCD Engine consists of a graph transitivity based algorithm that extracts CSCD relationships between concepts in the knowledge units. The overall procedure for CSCD Engine is shown in algorithm 1 in Fig. 4.

Domain Lexicon: The Domain Lexicon contains concepts that are related to computer sciences and can save it as base knowledge and update it during the system process.

 ISBN: 978-1-61208-518-0

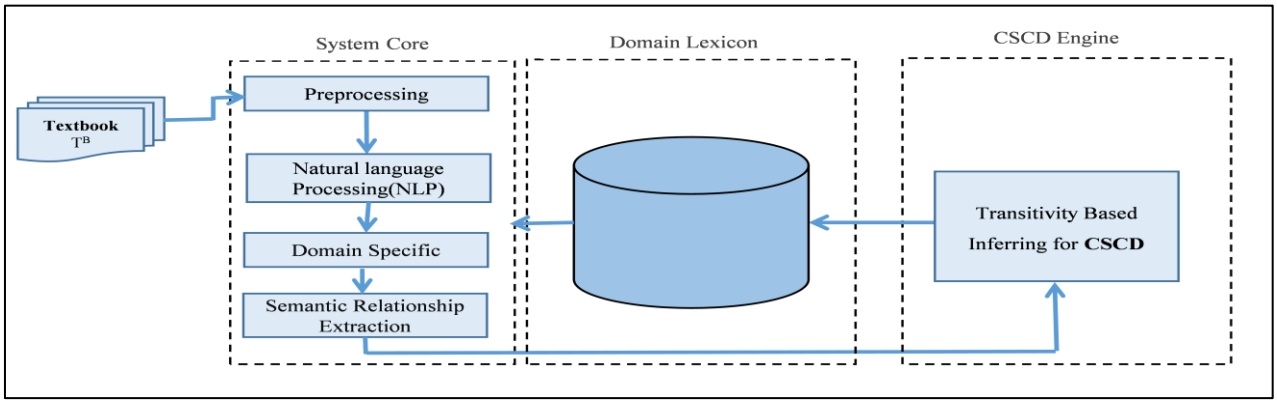


Figure 2. Overview of the System.

V. GRAPH TRANSITIVITY BASED TECHNIQUE

The System Core component plays a very important role in our system. It is the first step for preprocessing the textbook. The output from this component is a semantic graph G^S. In *CSCD* Engine, transitivity based technique used to infer *CSCD* levels from G^S. According to the linguistic view, the connection between words in a sentence is represented by the verb. Verbs, are hypothesized to indicate semantic relations between concepts. In this work the relationships between knowledge units indicated by verbs. We applied the transitivity technique to infer the *CSCD* levels between knowledge units as it is known to us that *CSCD* levels are divided into four levels based on verb majority [14].

The cognitive graph G^c consists of nodes and edges, where nodes are a set of concepts and edges are a set of verbs. Each edge connects two concepts via a specific verb, with each edge having a type (e.g., *CSCD* level). Multiple links between the same pair of concepts are possible. In our cognitive graph, the meaning of an edge between any two nodes is the *CSCD* level. The concept of transitivity between three nodes (c_i, c_j, and c_k) is defined as, if a node c_i has a link to node c_j and node c_j has a link to node c_k, then a measure of transitivity in the graph is the probability that node c_i has a link to node c_k. In general, we refer to c_j as a neighbor of c_i if c_i and c_j are directly connected in the graph. We also refer to the degree of a node as the number of neighbors it has. Fig. 3 shows the transitivity cases.

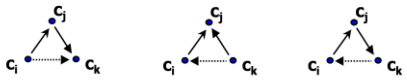


Figure 3. Transitivity Cases.

Based on the assumption that our graph G^S is transitive, we define the transitivity as a path with three hubs length to represent the relationship between two concepts and infer the hidden one. It is represented as three edges that connect a concept ci with a concept cj in the graph, where ci and cj are concepts from a knowledge unit. The edge between any two concepts in the path is one of the *CSCD* levels, which include {Understanding, Analysis, Applying, and Evaluating, and Creating}.

Algorithm 1 in Fig. 4 represents the graph transitivity technique as follows: the algorithm for transitivity has been implemented using Python programming language, providing a solid foundation to use a variety of NLP packages such as NLTK and NetworkX for graph operation. It starts with a source node which represents a knowledge unit (KU) in G^c; the algorithm then initializes the transitivity list and set of concepts or KU.

Algorithm 1: Transitivity Based Technique
Input: Semantic Graph G^S
Output: Transitivity Relationships between C
Def ExtractTransivityFromGraph(self):

```
1.  Transitivity= [ ]
2.  Concepts(C)=set ( )
3.  For each(C) in Graph(self):
4.  C.add(C)
5.  C.Nighbour=set()
6.  Nighbours=set(self. Graph [n])
7.  For Neighbor in Neighbor's:
8.  If Nighbour in Concepts:
9.  continue
10. Nodes_Nighbour.add(Nighbour)
11. For Nighbour_of_Nighbour in
    Nighbours.intersection( self. Graph [Nighbour]):
12. Nlist[].append(C[0],count)
13. If Nighbour_of_Nighbour in Nodes or Nighbour_of_Nighbour
    in Nodes_Nighbour:
14. continue
15. Transitive. Append((n,Nighbour,Nighbour_of_Nighbour) )
16. Return Transitivity (G^B)
```

Figure 4. Graph Transitivity Technique.

VI. KNOWLEDGE UNITS CLASSIFICATION

We classify the knowledge units based on the CSCD levels $\beta i = \{B_1, B_2, B_3, B_4\}$ as well as the relations between concepts in knowledge units. The transitivity technique discovers the hidden connections between concepts, and how those concepts are connected to a given knowledge unit, it also investigates the association among knowledge units themselves. The technique presents a strong connectivity between the concepts and knowledge unit.

Transitivity classifications are the sub-graphs extracted from G^C based on the transitivity relationships in the graph. What we have done is try to understand the relationships, which are the prerequisite relationships between concepts, by analyzing the graph. Our classification is divided into four classes as follows:

 ISBN: 978-1-61208-518-0

- C^{ST} (Strong Transitivity): let t denote the target node or knowledge unit, which is shared multi transitivity with their direct neighbors. This class classifies concepts into one of the *CSCD* levels, which is the Creation level. The connectivity between concepts is represented by one of the verbs in Bloom's measurable verbs list [14]. In addition, concepts, which are connected to the knowledge unit are strongly related to it. The concepts must be mastered if the learner needs to learn this knowledge unit. Fig. 5 represents the concepts' transitivity with t i.e., C^{ST}= {tA, tB, tC}.
- C^{MT} (Multi Transitivity): let t denote the target node or knowledge unit; this class consists of the neighbors of t, which shared multi transitivity with t and t's neighbors. The concepts in this class represent another *CSCD* level, which is the Evaluation level. Transitivity relationships among knowledge units in this class overlap and require judgment based on some criteria for some knowledge units. Fig. 5 represents the concepts in this class i.e., C^{MT}= {tE, tD, tW, tZ}.
- C^{WT} (Weak Transitivity): assume t denotes the target node or knowledge unit and it does not share any transitivity relationship with its direct neighbors but their neighbors do share transitivity relationships with other neighbors. Fig. 5 represents the concepts in this class i.e., C^{WC}= {tI, tH, tG}. The concepts in this class represent one of the *CSCD* levels, which is the Applying and Analyzing level. The transitivity relationship inferred combinations of concepts that represent framework to each other.
- C^{DT} (Disconnected Transitivity): concepts in this class are not sharing any transitivity with t or with its neighbors. Actually, the concepts represent the lowest level of *CSCD* levels, which is the Understanding and Remembering level. Most of the concepts are common and not related to the domain under study. Fig. 8 represents the concepts in this class i.e., C^{WC}= {O, P, Q, R, S, V}.

VII. EXAMPLE OF THE PROPOSED TECHNIQUE

The proposed technique goal is to classify the knowledge units CSCD levels in any given text. For example, consider a knowledge unit (topic) in an Algorithm textbook talking about Quick-Sort Algorithm; we need to classify the knowledge unit into CSCD levels. This section will start explaining our technique through this knowledge unit. Fig. 6 explains the knowledge unit from a textbook.

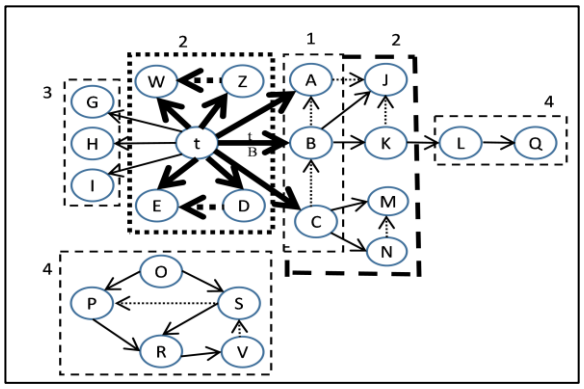
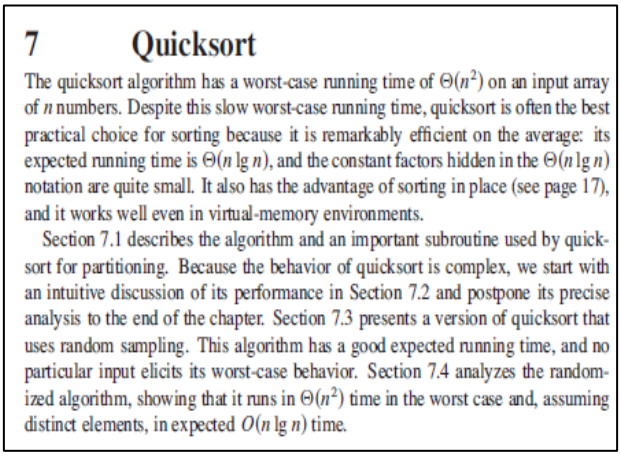


Figure 5. Graph Transitivity Classes.



7 Quicksort

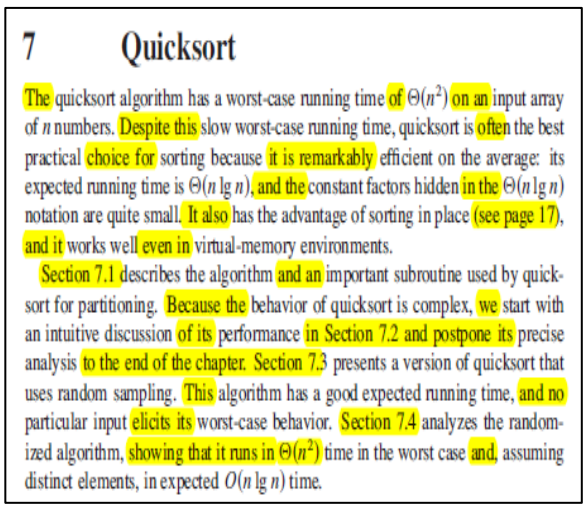
The quicksort algorithm has a worst-case running time of $\Theta(n^2)$ on an input array of n numbers. Despite this slow worst-case running time, quicksort is often the best practical choice for sorting because it is remarkably efficient on the average: its expected running time is $\Theta(n \lg n)$, and the constant factors hidden in the $\Theta(n \lg n)$ notation are quite small. It also has the advantage of sorting in place (see page 17), and it works well even in virtual-memory environments.

Section 7.1 describes the algorithm and an important subroutine used by quicksort for partitioning. Because the behavior of quicksort is complex, we start with an intuitive discussion of its performance in Section 7.2 and postpone its precise analysis to the end of the chapter. Section 7.3 presents a version of quicksort that uses random sampling. This algorithm has a good expected running time, and no particular input elicits its worst-case behavior. Section 7.4 analyzes the randomized algorithm, showing that it runs in $\Theta(n^2)$ time in the worst case and, assuming distinct elements, in expected $O(n \lg n)$ time.

Figure 6. A KU from a Textbook.

First, we start with the preprocessing of the given knowledge unit. The output is in Fig. 7; the yellow words which are stop-words were removed from the KU. After that, the *System Core* component extracts the relations among the concepts in a knowledge unit. The output is a semantic graph G^S presented in Fig. 8 where the figure explains all the possible relationships in the given knowledge unit levels for the analyzed knowledge unit, which is a Quick-Sort. It also includes a set of color codes to be used for our classification categories.

In the semantic graph, Fig. 8, we check the transitivity between concepts. First, we checked the relationship type between each two concepts, which is represented by the verb. Based on that, we classified the concepts in each knowledge unit into CSCD levels, and then we classified the knowledge units themselves. Fig. 9 demonstrates all the concepts classified into CSCD levels for the analyzed knowledge unit, which is a Quick-Sort.



7 Quicksort

The quicksort algorithm has a worst-case running time of $\Theta(n^2)$ on an input array of n numbers. Despite this slow worst-case running time, quicksort is often the best practical choice for sorting because it is remarkably efficient on the average: its expected running time is $\Theta(n \lg n)$, and the constant factors hidden in the $\Theta(n \lg n)$ notation are quite small. It also has the advantage of sorting in place (see page 17), and it works well even in virtual-memory environments.

Section 7.1 describes the algorithm and an important subroutine used by quicksort for partitioning. Because the behavior of quicksort is complex, we start with an intuitive discussion of its performance in Section 7.2 and postpone its precise analysis to the end of the chapter. Section 7.3 presents a version of quicksort that uses random sampling. This algorithm has a good expected running time, and no particular input elicits its worst-case behavior. Section 7.4 analyzes the randomized algorithm, showing that it runs in $\Theta(n^2)$ time in the worst case and, assuming distinct elements, in expected $O(n \lg n)$ time.

Figure 7. Text Preprocessing of the KU.

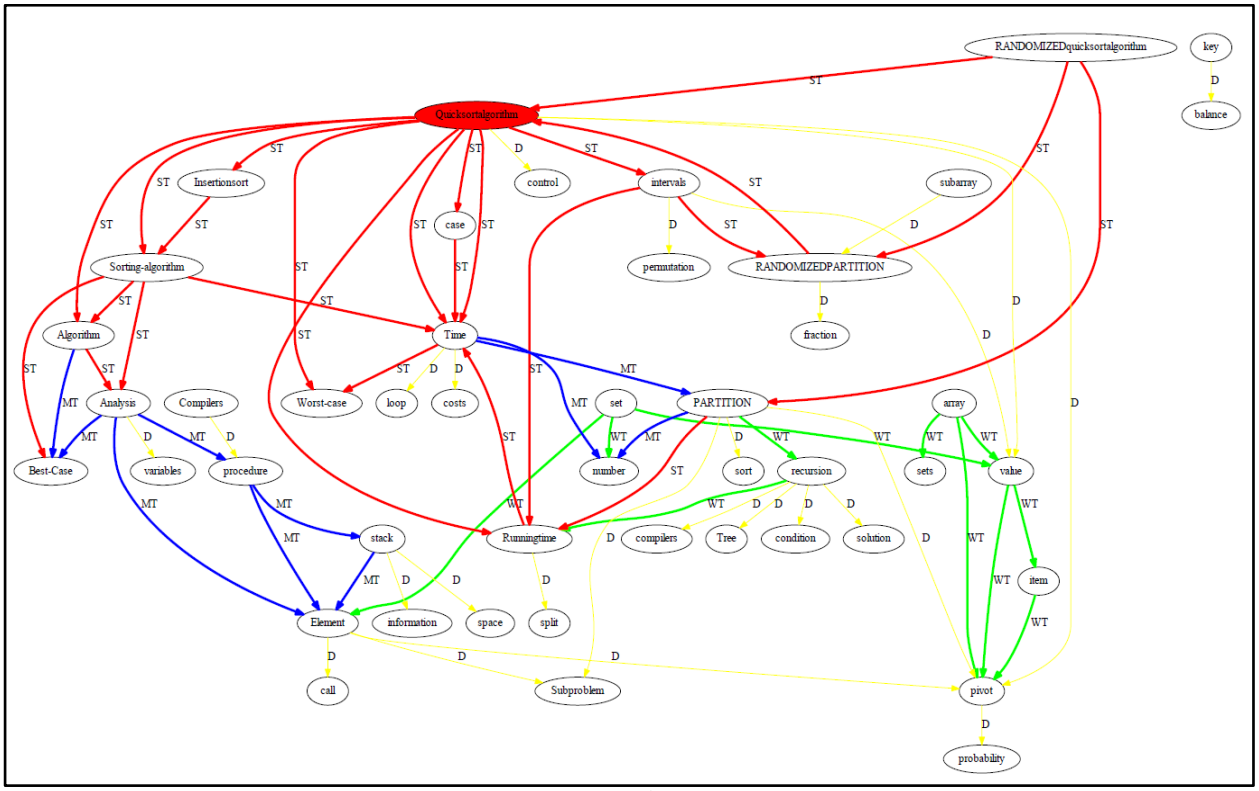


Figure 8. Semantic Graph Gs for a Knowledge Unit.

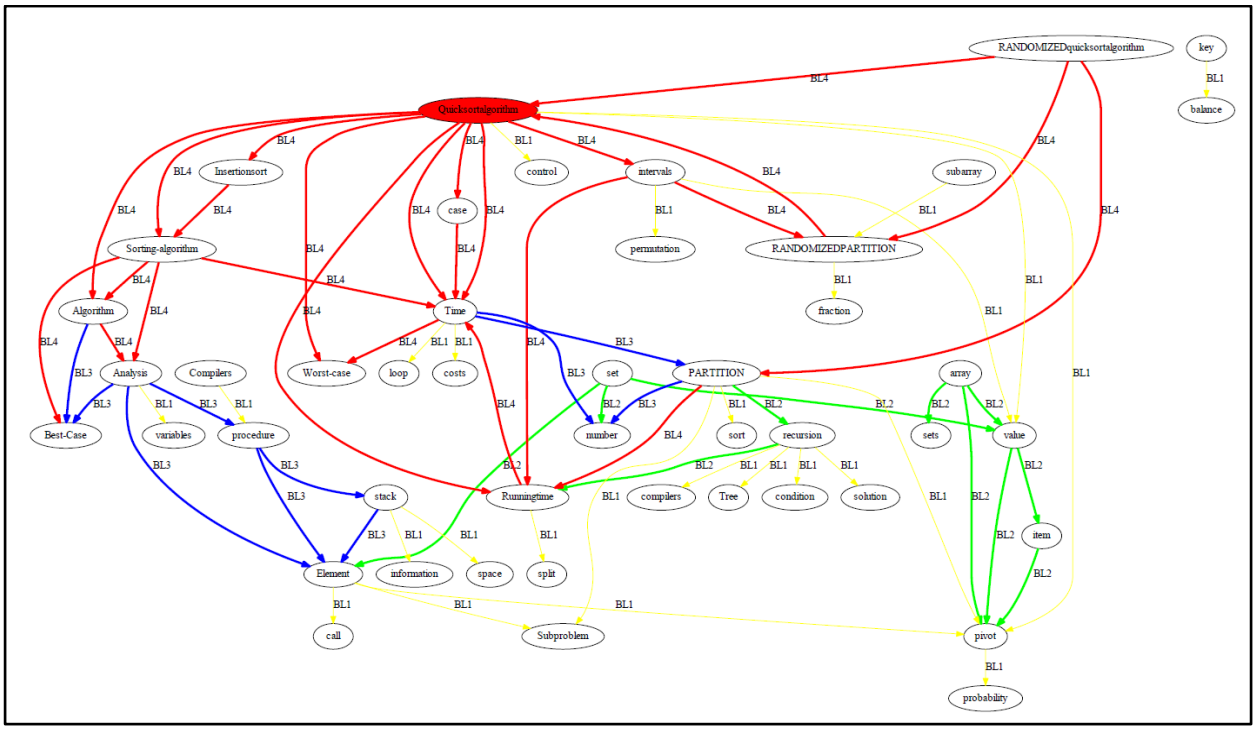


Figure 9. Cognitive Graph Gs for a Knowledge Unit.

VIII. EXPERIMENT SETUP AND EVALUATION

In this section, we first discuss the data set used for testing our system, and then the evaluation metrics will be presented. We used the same data set used in our previous work [3] to see the result from graph transitivity view.

A. Experiment Setup

We test the technique in two ways: locally and globally. Locally means the behavior of the technique using a knowledge unit from the same textbook. Globally means using three high quality textbooks that are used in computer sciences classes as course materials. We apply our technique to see how it performs on textbooks. We obtain three collections of documents from three textbooks on "Introduction to Algorithms" and "Data Structures and Algorithms" and "Algorithms", respectively. The textbooks are used as textbook for computer sciences courses at many universities. Table I shows the statistical information about the three textbooks.

TABLE I. PHYSICAL CHARACTERISTICS OF THE TEXTBOOKS

	Book1	***Book2***	***Book3***
TOC depth	4	3	2
Number of Knowledge unit	120	60	30
Number of extracted Relationships	8500	8200	3000
Number of concepts	1060	1020	950
Number of verbs	610	480	300
Overlaps of Knowledge units	400	300	220

We start with the preprocessing of the text which is the most important step as it includes the stop word filtration, to save only the domain specific concept. Then the semantic relationships extractor is used to extract the semantic graph and we classify all concepts within knowledge units. These knowledge units themselves are classified into CSCD levels which help reorganize the textbook based on the cognitive skills to know at which cognitive level each knowledge unit must be given for learners.

In this experiment, we used Introduction to Algorithm book that includes different knowledge units (topics). Fig. 10 presents the transitivity distribution of concepts in knowledge unit #1 which represented a Heap Sort topic from the textbook. It can be clearly seen that the number of transitivity is high for the concepts which are strongly related to the knowledge unit; the rest of the concepts, which have no transitivity, are common concepts and help knowledge units connect to each other.

Fig. 11 shows the graph connectivity measures which are: *Degree Centrality* **ω** and *Betweennes Centrality* **γ** for concepts in KU#1. Both measures prove how strong the concepts related to the knowledge unit are. It means the concepts that appear at the beginning of the chart are concepts related to the domain under study while the common concepts come at the end of the chart.

Additionally, Fig. 11 shows the graph connectivity measures which are: *Clustering Coefficient* (**φ**) and *Eigenvector Centrality* (**μ**) for concepts in KU#1. Both measures prove how strongly related the concepts are to the knowledge unit. It means the concepts that appear at the beginning of the chart are concepts related to the domain under study while the common concepts come at the end of the chart.

B. Graph Connectivity Measures

At this step, to measure the concept and knowledge unit connectivity which could be correlated with the graph metrics, we collected and calculated the following success measures:

Clustering Coefficient (**φ**): it is the measurement that shows the connectivity among knowledge units and the concepts related to them. According to [16] the mathematical formula of **φ** is as follows:

$$\varphi_i = \frac{2e}{k(k-1)} \quad (1)$$

Where *i* is a knowledge unit with degree *deg (i)* = *k* in G^T φ_i Takes values as $0 \leq \varphi_i \leq 1$

Degree Centrality (**ω**): as in [17], it shows that the interactions of a target concept are represented in the knowledge unit with other concepts in G^T. Our result shows the high centrality of the target concept in G^T. ω is defined as in question 2.

$$\omega_i = \deg(i) \quad (2)$$

Betweenness Centrality (**γ**) illustrates the connectivity between the target concepts and their neighbors by making a path between concepts; that is calculated as follows [24]:

$$\gamma(w) = \sum_{(i,j)\in V(w)} \frac{\sigma_{ij}(w)}{\sigma_{ij}} \quad (3)$$

Eigenvector Centrality (**μ**) as in [17] presents the importance of the target concept's neighbors which measure how well-connected a knowledge unit is to other highly connected concepts in G^T.

C. Evaluation

In order to evaluate the quality of the G^C for each knowledge unit, we are interested in two different measures. The first one expresses the completeness of the set of CSCD relationships, that is, how many valid CSCD relationships are found with respect to the total number of extracted relationships.

The second measure indicates the reliability of the set of CSCD relationships found in the knowledge unit, that is, how many valid Bloom relationships are

found with respect to the total number of CSCD in the knowledge units.

To compute the metrics, we compare our system with ground truth by asking Ph.D. students using their own background knowledge and additional resources to classify some knowledge units from a textbook and determine the level of the cognitive skills for each knowledge unit. The students created a semantic graph for each knowledge unit, so for each graph, we perform the ground truth in three knowledge units from the textbook. The purpose is to create a final classification G^C for each knowledge unit as similar as possible to the automatic system.

TABLE iI. PHYSICAL CHARACTERISTICS OF THE TEXTBOOKS

Book Name	# Knowledge Unit	Bloom Trajectory for KU
Introduction to Algorithm	35	90
Algorithms	10	40
Data structure and Algorithms	30	25

IX. CONCLUSION AND FUTURE WORK

In this paper, an automatic technique that finds the relationships between knowledge units according to *CSCD* levels has been presented. The technique is an improved version of our previous work [3]. Discovering relationships based on CSCD levels is a novel and challenging problem. The results show that the relationships between knowledge units are different from one textbook to another. Based on our analytical result, it is possible to conclude that by using CSCD levels we can decide which parts of a textbook to use at which level of learning to match the learner's skills. For future research, we will investigate the use of the method to evaluate online learning resources.

ACKNOWLEDGMENTS

We take this opportunity to thank all the reviewers for this paper for the suggestions that provide helpful tips to improve the paper.

REFRENCES

[1] B. Benjamin Samuel. "Taxonomy of educational objectives". Vol. 2. New York: Longmans, Green, 1964.

[2] A. Lorin W., David R. Krathwohl, and Benjamin Samuel Bloom. "A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives". Allyn & Bacon, 2001.

[3] N. Fatema and Khan J."Conceptualize the Domain Knowledge Space in the Light of Cognitive Skills". In Proceedings of the 7th International Conference on Computer Supported Education. 2015.

[4] P. Machanick,"Experience of applying Bloom's Taxonomy in three courses". In Proc. Southern African Computer Lecturers' Association Conference, 2000.

[5] R .Lister and J .Leaney, "Introductory programming, criterion-referencing, and bloom". In ACM SIGCSE Bulletin.2003

[6] D.Oliver, et al. "This course has a Bloom Rating of 3.9." Proceedings of the Sixth Australasian Conference on Computing Education-Volume 30. Australian Computer Society, Inc., 2004.

[7] H. Marti, "Automatic acquisition of hyponyms from large text corpora." Proceedings of the 14th conference on Computational linguistics-Volume 2. Association for Computational Linguistics, 1992.

[8] R. Alan, S. Soderland, and O. Etzioni. "What Is This, Anyway: Automatic Hypernym Discovery." AAAI Spring Symposium: Learning by Reading and Learning to Read. 2009.

[9] F. Frédéric, and Francky Trichet. "Axiom-based ontology matching." Proceedings of the 3rd international conference on Knowledge capture. ACM, 2005.

[10] R. Kanagasabai, and A. Tan. "Mining semantic networks for knowledge discovery." Data Mining, 2003. ICDM 2003. Third IEEE International Conference on. IEEE, 2003.

[11] N. Roberto, P. Velardi, and A. Gangemi. "Ontology learning and its application to automated terminology translation." IEEE Intelligent systems,2003.

[12] P. Mathew, Snehasis Mukhopadhyay, and Matthew Stephens. "Identification of biological relationships from text documents." Medical Informatics. Springer US, 2005.

[13] A. Mohammad, I. Barjasteh, and H. Radha. "Transitivity matrix of social network graphs." 2012 IEEE Statistical Signal Processing Workshop (SSP). IEEE, 2012.

[14] Bloom's Taxonomy Action Verbs.http://www.clemson.edu/assessment/assessment practices/referencematerials/documents/Blooms%20T axonomy% 20Action%20Verbs.pdf,2011.

[15] L. John, "Linguistic semantics: An introduction." Cambridge University Press, 1995.

[16] M. Newman,"A measure of betweenness centrality based on random walks." Social networks,2005.

[17] A . Réka, H. Jeong, and A. Barabási. "Error and attack tolerance of complex networks." 2000.

[18] H. Thomas, E. Charles, L .Ronald, and C. Stein."Introduction to Algorithms, Third Edition (3rd ed.)". The MIT Press. 2009.

 ISBN: 978-1-61208-518-0

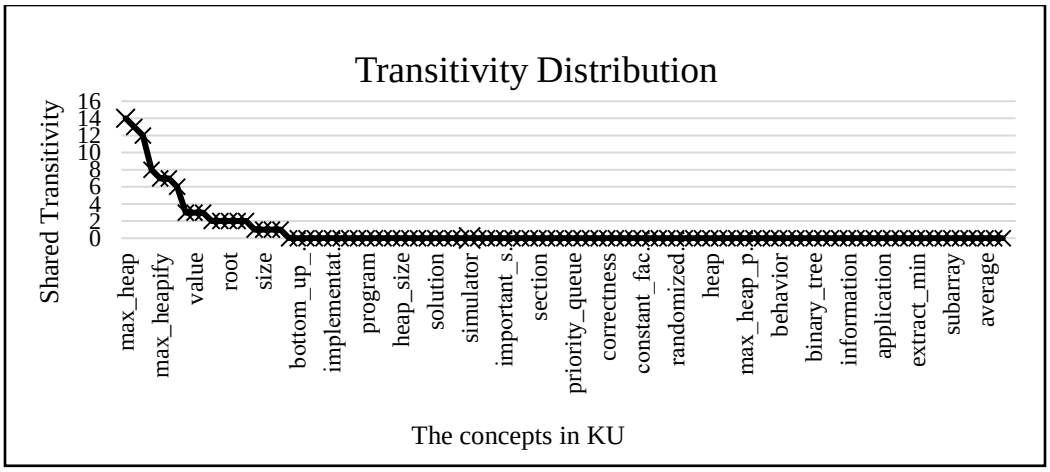


Figure 10. Tringluarity Distribution for KU #1(Heap Sort).

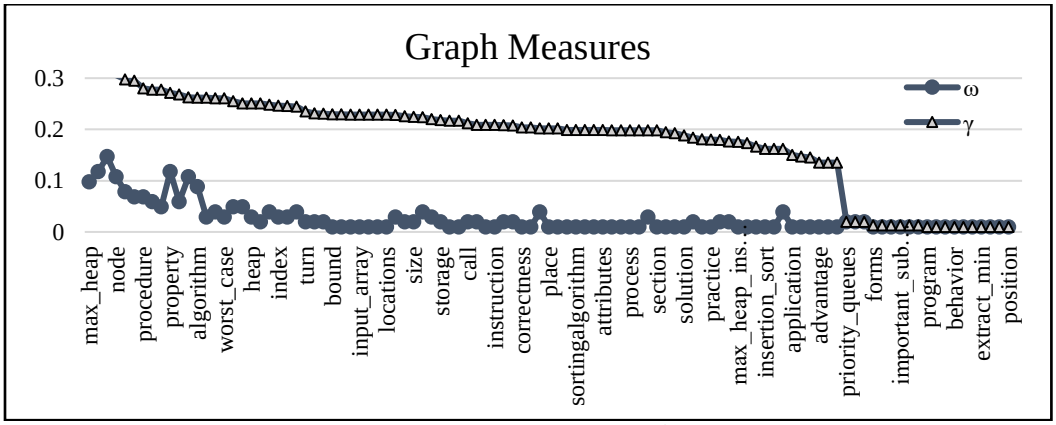


Figure 11. Graph Connectivity Measures for KU #1.

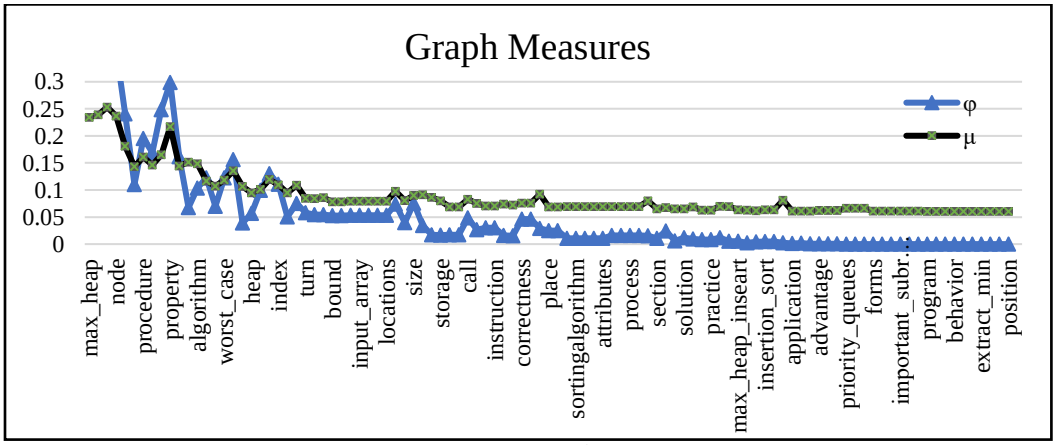


Figure 12. Graph Connectivity Measures for KU #1, KU#2, and KU#3