Applying Data Organizational Techniques to Enhance Air Force Learning

THESIS

Jacob A. Q. Orner, Second Lieutenant, USAF

AFIT-ENG-MS-20-M-052

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-MS-20-M-052

Applying Data Organizational Techniques to Enhance Air Force Learning

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Cyber Operations

Jacob A. Q. Orner, B.S.C.S.

Second Lieutenant, USAF

March 19, 2020

AFIT-ENG-MS-20-M-052

Applying Data Organizational Techniques to Enhance Air Force Learning

THESIS

Jacob A. Q. Orner, B.S.C.S.
Second Lieutenant, USAF

Committee Membership:

Richard Dill, Ph.D
Chair

George E. Noel, Ph.D
Member

David S. Long, Ph.D
Member

Mark G. Reith, Ph.D
Member

AFIT-ENG-MS-20-M-052

Abstract

The United States Air Force (USAF) and the Department of Defense (DoD) use traditional schoolhouses to educate and train personnel. This method limits the number of students due to classroom size, funding, and faculty availability. One method to increase throughput is to decrease the length of training to create an opportunity to graduate more students within a given time frame. However, while cutting material could graduate more students, it may not be the optimal solution. Another method is to shift towards an asynchronous learning environment in which students move through content at individual paces and paths. This research introduces a methodology for transforming a set of unstructured documents into an organized Topic Map (TM) students can use to orient themselves in an education domain. The research then identifies different learning paths within the TM to create a directed Knowledge, Skills, and Abilities Tree (KSAT) paralleling the National Initiative for Cybersecurity Education (NICE) Knowledge, Skills, and Abilities (KSA) construct. We apply this methodology in four case studies, each of which is an education or training course, in the results section. The results were validated via visual inspection and compared to Subject-Matter Expert (SME) created TMs using a validated graph comparison metric. The outputs of this research suggest this methodology can be used to supplement the Air Force (AF) education system by mapping topics and course material for students to ingest prior to beginning a course, during a course, or after completion to refresh their knowledge. The research produced three TMs and a KSAT for all four case studies and modified KSATs for two of the studies where modular learning paths were identified. Using a graph comparison metric and the topic identification rates for the TMs, we tested a whitelisting algorithm to identify topics with up to 81%

accuracy, and leveraged a Latent-Dirichlet Allocation (LDA) algorithm on its own and the same LDA algorithm with a modern ontological system for lexical knowledge in ConceptNet. The results of this research show that TMs and KSATs can automatically be created with minimal user input. The methodology developed in this research could help the Air Force increase the amount of airmen progressing through education and training pipelines.

Acknowledgements

I owe so much of my accomplishments today to those around me. To my family, friends, and especially to Molly, thank you for all your support; the visits, edits, reviews, and care packages mean the world to me.

Jacob A. Q. Orner

Table of Contents

Page

List of Figures

List of Tables

Applying Data Organizational Techniques to Enhance Air Force Learning

I. Introduction

1.1 Motivation

For the Air Force (AF) to meet the demand to produce cyber Airmen, a reliance on automation and Natural Language Processing (NLP) techniques to develop material and assess Airmen competencies efficiently should be expected [2]. As of 2019, Air Education and Training Command (AETC) has failed to meet cyber manpower demands to fill Cyber Protection Teams (CPTs) and Mission Defense Teams (MDTs), creating 555 cyber warriors per fiscal year compared to the Air Force cyber skill demand being as high as 3,000 per fiscal year [2]. The complex mission requirements demand Airmen possess a robust and tailored skill set and requires an adaptive student-centered learning approach [3].

The current method of relying on traditional brick and mortar training does not consider the students current skill, ability, and aptitude, but puts every cyber student through the standard pipeline. While the standardization of education and training has benefits, the content and assessments students must pass can be standardized in a more modern system while still addressing the issues of cost and throughput. With an average travel and per diem cost of $700 round trip from home station to school house, and $104/per day/student, the Air Force spends overs $100 million each year[1][2]. This learning model is expensive and does not exploit the 21st century advanced education techniques that adapt learning around the student. Instead, the Air Force, regardless of inherent skills or aptitude consistently and uniformly

[1]$115,980,000 = (365 days * $104 + $700) * 3000 students

progresses each students through serialized learning paths in a test, advance, repeat model. Additionally, some airmen wait on casual status for training slots to open, costing the Air Force more time and money [3]. These inefficiencies, coupled with an increasing need for cyber professionals of the same caliber in the Air Force and DoD, presents a training and education challenge.

Training shortfalls go beyond cyber. The Pilot Training Next [4] program aims to shorten waiting times before training pilots. This relies on virtual reality to decrease total time spent in pilot training by two-thirds, and costs by over half [5]. Additionally, AETC has launched the Continuum of Learning (CoL) initiative to consolidate the different learning pipelines into a centralized repository of data in the Air Force Learning Services Ecosystem (AFLSE) that can be updated, adjusted, and tailored towards specific goals with aim to streamline AF education and training.

In this thesis, we develop, test, and evaluate a methodology, in parallel with the goals of the AETC CoL, to produce content faster by organizing data for the warfighter to access. By leveraging NLP techniques to organize unstructured content into a Topic Map (TM), an undirected graph of nodes representing topics and edges representing a relationship between them, and provide a custom Knowledge, Skills, and Abilities Tree (KSAT), a directed TM with a second node type to provide direction along a learning path, we can improve the quality of the education and training material in self-guided learning platforms for Air Force members.

We demonstrate the effectiveness of the methodology in a tool: the Pedagogical Resource Organization Framework (PROF). PROF consumes textual data, identifies key topics, recognizes contextual relationships, and produces a TM automatically, and a KSAT with the additiona of two human-generated documents. This KSAT is displayed and evaluated to identify and highlight educational paths for students.

When applied to a dataset, PROF automatically generates a TM and KSAT.

These assist educators to develop courses and organize content. TMs provide an overview of topics in the data set and allows users to choose areas of interest by linking them to associated content. KSATs provide learning paths for users to progress through courses and build their Knowledge, Skills, and Abilities (KSA)[2].

This chapter provides the motivation, problem statement, research questions, research assumptions, and contributions of this research. Chapter II, explores related literature, defines terms as they pertain to this research, and identifies gaps in the field of study. Chapter III describes the methodology and the algorithms used to create the TMs and KSATs for each one of the case studies. Chapter IV discusses and analyzes the results of the research. Finally, Chapter V draws conclusions about the research, highlights contributions made to the field, and suggests areas of future work.

1.2 Problem Statement

To shift to asynchronous and self-guided learning environments, tools and processes that aid the development of training content should be researched and leveraged to make this process more efficient and easier for educators.

1.3 Research Questions

In this thesis, we propose the following research questions.

1. ***Investigative Question 1: How closely can computational semantic processing generate a Topic Map compared with a SME developed one using a predefined comparison metric?***

   Hypothesis: Using semantic analysis and a predefined methodology, a Topic Map

[2]The Department of Defense (DoD) uses knowledge, skills, and abilities to evaluate candidates' resumes applying for federal jobs [6].

can be created automatically from a data-set that is accurate between 0.1 and 0.5 using the McClure Validity Score [7].

2. ***Investigative Question 2: How much can modern ontological techniques improve topic generation?***

   Hypothesis: With SME-generated topics as the standard, Latent-Dirichlet Allocation can identify topics from a dataset. In addition, the introduction of an ontological techniques will maintain or increase this identification rates by up to 20% through topic word clustering.

3. ***Investigative Question 3: To what extent can we automate the creation of a KSAT for a course?***

   Hypothesis: We hypothesize that the methodology created in this research provides an almost automatic process for creating a KSAT for a course from its documentation that requires less than 30 minutes of working time.

1.4 Methodology

This research creates and tests a methodology for creating TMs and KSATs from unstructured data. This methodology identifies key topics and the relationships between these topics and creates a TM, then creates a KSAT by organizing these topics and relationships into a hierarchy of generalization or complexity. The hierarchy provides a learning path for students to follow to complete lesson objectives. PROF can assist educators in generating content and objectives for students to complete. TM and KSAT equivalents have been examined as education and training aids to improve student motivation and engagement for use on the Air Force Institute of Technology's Cyber Education Hub (CEH) [8], Kahn Academy [9], and other learning sites, but we are not aware of a comparable tool at this time for generating these automatically.

We evaluated PROF based on its ability to generate and organize topics and objectives on four Air Force courses: Air Force Institute of Technology (AFIT)'s Mobile and SCADA Security course (CSCE 660), its Agile Software Systems Engineering course (SENG 593), and its Model-Based Systems Engineering (MBSE) Course. The final case study evaluated is the 39th Information Operations Squadron's Network Warfare Bridge Course (NWBC). These case studies were chosen as they contain a range of length, content type, and education goals that can be used to draw insight into which algorithm and course type is best suited to the methodology.

The methodology leveraged three algorithms for testing. For the first, researchers created a whitelist, a document containing a list of possible topic candidates, using the learning objectives from the course. PROF uses frequency analysis to identify topics and relationships from course documents. The outcome is a course-specific TM. The second algorithm tested in phase one leverages Latent-Dirichlet Allocation (LDA), an NLP algorithm for automatically generating topics. The third algorithm leverages this same algorithm, but also incorporates ConceptNet, a lexical ontology, with the goal of more accurately naming topics in the TM. Phase two involved introducing the syllabus to the output of the whitelisting algorithm to provide direction and structure, thus creating a KSAT.

The course documents consisted of MS Word [10], MS PowerPoint [11], and Portable Document Format (PDF) [12] files, among others. Before processing, PROF uses text extraction techniques to convert all files into text files.

In creating each of the structures for the different courses, as well as testing different algorithms for each, a set of TMs and KSATs were produced for each course. This allowed the comparison of structures to a predefined metric using the McClure score, and provided additional insight into how education compare to training courses.

To validate the created TMs and KSATs, researchers asked a Subject-Matter

Expert (SME) for each case study to create a TM that depicts the case study's learning objectives and topics. PROF's TMs are compared against the SME TM using the McClure comparison scoring metric [7]. Researchers then visually inspect the KSATs to identify additional learning paths in the courses based on how nodes became clustered within the KSAT.

1.5 Assumptions

The following assumptions are made during this research.

1. TMs and KSATs are effective techniques for organizing and evaluating education concepts [13].
2. SME-Created TMs are the baseline for TM quality. While three SMEs could create different TMs, we assume a SME-Created TM is the standard [7][14].
3. Reducing the time needed for educators to create TMs and KSATs is beneficial to student learning and course development [15][16].
4. PROF assists educators in creating TMs and KSATs at a faster rate than manual creation.

1.6 Research Contributions

1. Define and demonstrate features that may be used in TM and KSAT generation.
2. Informs other researchers about techniques for evaluating synthetic TMs and KSATs.
3. Automatically generate topics with up to 80% accuracy to SME-generated topics.

4. Extract learning paths from organized TMs that can be used to modularize courses.

1.7 Conclusion

This experiment's intent is to test methods that automatically creating TMs and KSATs, and evaluate the output of each method against a baseline to evaluate which will be most suitable to identify topics, relationships, and learning paths. PROF, when applied to unorganized data repositories, effectively identifies topics and learning paths.

The next chapter discusses the background and literature review for this research where we identify gaps in the current knowledge base, adopt and define our terms, and demonstrate how our research fits in the state of the art.

II. Background and Literature Review

2.1 Education

Although the number of people with degrees and certifications in the computer science career field has increased in the last decade, there is still a workforce shortage due to the Knowledge, Skills, and Abilities (KSA) focused on in eucation not matching those emphasized in the field [17]. This dissonance in an increase in paperwork for members of the field and a stagnant ability level can be attributed to differing levels of quality, depth, and applicability of these programs [18]. To improve skill retention, the Department of Defense (DoD) and private companies have developed requirements for education and training in the cyber and information technology career fields [19]. With the DoD and Air Force implementing these requirements, the burden falls on educators, trainers, and supervisors to create relevant, up-to-date courses. We explored the field for tools, techniques, and processes for mapping courses to identify and validate their lesson objectives against requirements to ensure they are met. We identified gaps that could be explored and addressed with the research and the Pedagogical Resource Organization Framework (PROF) methodology. The tools we explored and the gaps we identified are discussed in several domains: traditional and asynchronous classroom environments;research into educational tools like a Topic Map (TM) and Knowledge, Skills, and Abilities Tree (KSAT); Natural Language Processing (NLP) basics; and graph comparison metrics.

2.1.1 Traditional Classroom Education

For the purposes of this research, a traditional classroom is an instructor-centered, synchronous learning environment where specific lessons are taught at each meeting, and students progress through the course at the same pace. The DoD has typically

conducted training and education in this traditional way since its inception [20]. The traditional classrooms have long been an efficient and popular way to host courses, and have been successful in bringing content and education to students [21]. However, courses of this nature do not allow students to move at different paces, regardless of prior knowledge, motivation, or ability to move at a faster pace, or difficulty to understand content. Those courses follow a singular track, with no opportunity for students to explore other relevant areas of interest.

2.1.2 Asynchronous Learning

Asynchronous learning environments are those where learners can access remote learning resources asynchronously and use self-study techniques with asynchronous interactivity to progress through a course [15]. Asynchronous learning environments in classroom education have been shown to be effective at garnering discussion of complex topics by giving students time to reflect and construct their ideas before contributing [22]. We expand the definition of asynchronous learning to include "flipped classroom" education, where students watching the course lecture on their own time and work on assignments, problems, and projects during scheduled class time. Ideally, students are able to work at a higher Bloom's taxonomy [23] level in class, and issues with content can be addressed at a more personal level between teacher and student [16]. This method of learning is more common in higher education. Asynchronous classrooms offer institutions a cost-effective way to educate students while educators restructure courses to focus on collaborative learning and deeper discussion about the content [24]. A study by Carswell, et al found students who took asynchronous courses demonstrated a statistically improved understanding of course material than those who took the synchronous course [25]. This increased efficiency and opportunity for deeper learning shows promise for a move towards asynchronous education

and training within the Air Force.

In asynchronous learning environments, educators use multi-modal learning techniques – using different education mediums (text, audio, visual, etc.) – to teach a concept [26]. Studies by Hazari and then Fadel have shown that this multi-modal approach provides superior results in students performance and learning compared to the traditional classroom method [27][28]. A conclusion Fadel stated in research demonstrates this point, saying "students engaged in learning that incorporates multi-modal designs, on average, outperform students who learn using traditional approaches with single modes" [27]. Martin's research showed that the approach non-technical people use to learn technical knowledge, skills, and abilities, is almost exclusively through search engines, which return these multi-modal forms of content in text, audio, and video formats. Students of today follow this approach when learning, and educators must account for this by expanding to a multi-modal array of content available to students for learning [29]. As courses shift to asynchronous environments and educators restructure them to be more efficient for students and educators a process for identifying which areas of course content relate closely to specific lesson objectives is required. Identifying these relationships allows educators to ensure each objective is addressed and provides students with multi-modal content to best suit their learning style.

2.2 Educational Tools

To properly identify an area of research which has not yet been addressed, we surveyed the field of related works in TM and KSAT development. In addition, we discuss the various definitions used in this field of study – researchers invoke different terms to define similar concepts. As this can be confusing, we adopted and created definitions to provide a baseline knowledge for the reader.

2.2.1 Topic Maps

Hatzigaidas et al enumerated several tools which support TM creation. They defined the basic concepts of a TM as Topics, Associations, and Occurrences; Topics represent a subject or kind of type, Associations are relationships between Topics, and Occurrences refer to the number of times a Topic is present in a specific document. They also described an intuitive graphical user interface to edit a TM. While surveying the different tools available for the creation, organization, and editing of TMs, Hatzigaidas et al identified a gap in the open-source tools available for creating TMs for large and/or complex data sets [30]. This shows the need for a TM tool that does not limit the amount of documents or topics included in a TM.

Alam et al compared open source tools and application programming interfaces (APIs) available for the storage, editing, browsing, and display of TMs and resource description frameworks. They noted that TM tools were less mature and lacked continued support [31]. Hatzigaidas and Alam's research showed a need for a comprehensive TM tool capable of handling all portions of the TM creation and updating process.

Research supports automatic creation and measurement of TMs. Zubrinic et al proposed a method for creating TMs automatically using a morphologically rich language, such as Croatian [32]. They proposed multiple different methods for creating TMs such as dictionaries, linguistic tools, structured data sources, and non-text data sources. Villalon & Calvo described methods for evaluating an automatically created TM against manually generated TMs to test a tool's effectiveness [33]. These methods use Subject-Matter Expert (SME) created TMs as the gold standard, and demonstrate a technique for measuring differences between SME-created and automatically created TMs. McClure et al defined this assessment method, which involves finding similar topics in each map, and dividing the intersection of the set of each

topic's neighbors by the union of the set of each topic's neighbors. This creates a value of similarity between zero and one. This is done for all like nodes in the two maps and averaged to give an overall score for the map [7][13]. Figure 1 depicts this formula, where i is a specific shared node, $C_{i,1}$ is the set of relationships of node i in Topic Map 1 (TM1), $C_{i,2}$ is the set of relationships of node i in Topic Map 2 (TM2), and A is the number of total shared nodes between TM1 and TM2.

$$\frac{\sum_{i=1}^{A}(\frac{|C_{i,1}\cap C_{i,2}|}{|C_{i,1}\cup C_{i,2}|})}{A} \tag{1}$$

Figure 1: McClure Formula for Calculating Topic Map Similarity

Vodovozov and Raud applied TMs to electrical engineering courses at the Tallinn University of Technology. They explored TMs as a method to identify connections between topics for both students and teachers; they found improvement in student achievement of learning outcomes when TMs were integrated into a course [14].

In another application, Letassy et al spent two academic years identifying KSA statements for creating the University of Oklahoma College of Pharmacy's curriculum to target learning outcomes through resequencing courses and defining expected ability levels at each year point in the program. This resequencing was shown to better match and teach the KSAs cited in the curriculum to the coursework students performed [34].

Another area of previous research tested creating TMs for assessment, test creation, and verification. Lin et al created TMs that included a course's key topics. They then created assessments which tested each of these topics and provided students with a TM highlighting the areas where they did not demonstrate knowledge. Lin et al showed that students using this system made statistically significant improvements in learning achievement compared to those who did not [35]. This process of

using TMs to highlight course learning objectives, identify areas of weakness, and adjust content accordingly could be useful to students and educators alike, but there is currently no tool to perform this process.

2.2.2 Knowledge, Skills, and Abilities

The National Institute of Standards and Technology (NIST) created a special publication, the National Initiative for Cybersecurity Education Cybersecurity Workforce Framework (NICE Framework) [1]. The NICE Framework details the interdisciplinary nature of cybersecurity work and the KSAs needed to complete tasks in the cybersecurity domain. The framework also standardizes terms within the cybersecurity education field to allow for easier communication between organizations, scholars, and sectors in the field [1]. The NICE Framework establishes the following definitions:

- **Knowledge** is a body of information applied directly to the performance of a function.
- **Skill** is an observable competence to perform a learned psychomotor act. Skills in the psychomotor domain describe the ability to physically manipulate a tool or instrument like a hand or a hammer. Skills needed for cybersecurity rely less on physical manipulation of tools and instruments and more on applying tools, frameworks, processes, and controls that have an impact on the cybersecurity posture of an organization or individual.
- **Ability** is competence to perform an observable behavior or a behavior that results in an observable product.
- **Task** is a specific defined piece of work that, combined with other identified Tasks, composes the work in a specific specialty area or work role [1].

For the purpose of this research, we adopt the NICE Framework definitions to ease understanding of the background, methodology, and results of the research. Figure 2 shows how these terms relate to each other, work roles, specialties, and categories or topic areas.

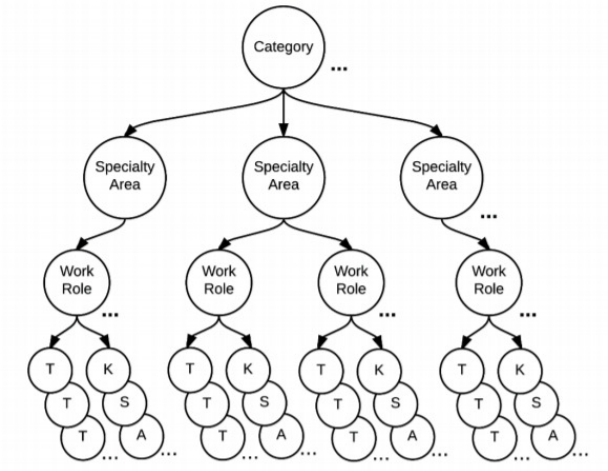

Figure 2: Relationships among NICE Framework Components, where Work Roles are composed of Tasks (T), Knowledge (K), Skills (S), and Abilities (A) [1]

In defining these terms and their relationships, we can scope the research to target specific goals and use the baseline set by the NICE Framework as a starting point for outlining and testing the output of the research.

2.2.2.1 Competencies

To establish KSATs that reflect the structure of educational classes, identification of core competencies is necessary. Kang and Ritzhaupt identified a framework from which these competencies can be identified and defined [36]. They invoked Norman's Know-Can-Do hierarchy to explain how to define, relate, and invoke Knowledge, Skills, and Abilities in various situations. Their research provides a framework to align identified competencies in a given field with the different levels of a KSA structure.

Kay and Moncarz's research to identify competencies in the lodging management

field proved successful. They defined and grouped competencies agreed upon by a SME and found that although one was rated as less important by experienced personnel, this competency proved most significant to differentiate success levels of workers [37]. This research showed that defining and identifying KSAs and their connections can provide insight into which areas are most important and how to apply effective learning methods to students to directly target those KSAs.

Research by van der Klink and Boon defined the relationships between Competencies and KSAs. Competencies do not equal KSAs, but are made up of combinations of the three categories of knowledge, skills, and abilities, depending on the specific competency [38]. Different combinations of the three in different competencies provides a method for users to build competencies from KSAs via tasks and activities. The distinction allows users to target different tasks and activities for a topic and effectively transition from knowledge to skills to abilities.

2.2.2.2 Itineraries

Cañas and Novak proposed a method to align TMs and KSAs using what they called "itineraries". Itineraries are TMs that guides users through the activities and tasks needed to build KSAs) to gain a competency. Cañas and Novak noted that TMs as course organizers are limited because a TM described a topic set clearly but did not show students how or in what order to learn the topics to demonstrate proficiency. Cañas and Novak's research provided a road map for the manual creation of itineraries and defined key terms used in this field. It also described an "itinerary of itineraries", or a way to make a course non-sequential by mapping the competencies, tasks, and knowledge needed for each learning outcome and linking them together to allow users to navigate through topics at their own discretion [39]. In addition, Cañas and Novak's research highlighted a gap in automation for this research.

2.2.3 Our Definitions

We found several synonyms throughout previous research, and now define four overarching terms.

Table 1: Definitions of keywords used in this research

Term	Synonyms in Literature	Our Definition
Topic	Learning Objective, Lesson Objective, Lesson Outcome, Subject	A subject or lesson objective within a course. Information that pertains to a specific subject area.
Topic Map (TM)	Knowledge Graph, Knowledge Map, Concept Map	An undirected graph consisting of nodes representing topics and edges representing a relationship between such topics.
Knowledge, Skills, and Abilities (KSAs)	Competencies	The content and proficiency associated with a topic, whether it be course readings, exercises, or assessments. Not visualized in a PROF TM but present in content associated with topic nodes.
Knowledge, Skills, and Abilities Tree (KSAT)	Skill Tree, Itinerary, Knowledge Tree	A directed graph consisting of topic and lesson nodes, with lesson nodes relating to topic nodes when topics are present within the lesson's content, and lesson nodes also possessing directed relationships with one another to represent the path of the course.

We defined a TM as an undirected graph which consists of nodes representing topics and the edges representing a relationship between such topics. Figure 3 is an example TM with five Topic nodes. Topic nodes may have zero (Topic E), one (Topic D), or more (Topics A, B, C) connections with other Topic nodes in the TM.

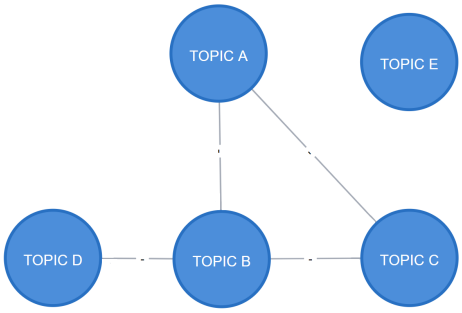

Figure 3: TM Node Example

Furthermore, a KSAT is defined as a TM with a second node type, the Lesson node, representing a degree of generalization in the subject or a lesson in the course. Adding the second node type gives the TM organized direction for students to follow. The direction moves from knowledge level to skill level to ability level. In parallel, the direction also moves from general to specific content, similar to traditional classroom methods of progressing through a course. The Lesson nodes represent nothing other than a numbered path the course follows.

Figure 4 shows how the different node types in a KSAT relate. The two nodes on the top are Topic nodes, and have an undirected relationship with one another. The two nodes on the bottom are Lesson nodes and are directed with one another to show course flow. The Lesson nodes provide a directed relationship through Topic nodes to reveal a potential path a student can travel to demonstrate learning. Lesson nodes may relate to one or more Topic nodes (bottom-left of Figure 4), and Topic nodes may be linked by one or more Lesson nodes (top-right of Figure 4).

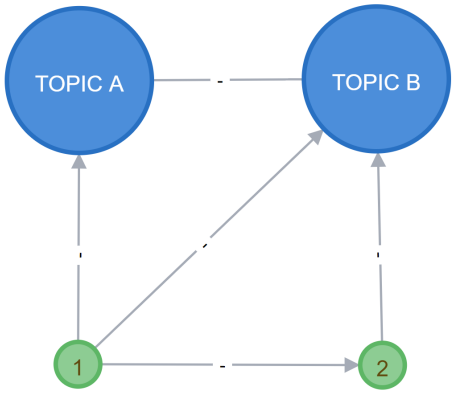

Figure 4: KSAT Node Example

2.3 Natural Language Processing

Statistical NLP can be defined as consisting of "all quantitative approaches to automated language processing, including probabilistic modeling, information theory, and linear algebra" [40]. To create TMs and KSATs from a repository of content like a training or education course, steps must be taken to identify relevant topics present in the data. Past research into TMs and KSATs found that manual creation of these structures took multiple hours, requiring domain knowledge of the depicted course [13]. We provide a discussion of frequency analysis, Latent Semantic Analysis (LSA), Probabilistic Latent Semantic Analysis (pLSA), and Latent-Dirichlet Allocation (LDA) [41][42][43][44][45].

2.3.1 Frequency Analysis

Frequency analysis is the mathematical process of identifying the number of unique words in a given document or set of documents and storing the number of times each word appears in the text divided by the overall number of words in the text. Depending on the corpus size and algorithm selected, analysis can require large amounts of preprocessing and computation. Research at the Xerox Palo Alto Research Cen-

ter proved the effectiveness of frequency analysis for document summarization and keyword identification [41].

To perform frequency analysis on a large corpus of text, the text is preprocessed to standardize the language and tense; this includes tokenizing and lemmatizing words and phrases in the documents. Tokenization is the process of breaking a set of characters or words into smaller units [46]. For instance, the sentence "He likes apples" would be tokenized into a list of the words "He", "likes", "apples". Lemmatization is the process of substituting a word with its etymological root [47]. For instance, the word "apples" would be replaced with apple. This yields a standardized dataset that supports NLP [40].

2.3.2 Topic Identification Algorithms

In this research, each document is preprocessed independently. From this document, the identification of topics occurs from frequency analysis. Unfortunately, the use of pronouns or synonyms may not be recognized due to a lack of context in the NLP process. This lack of context indicates that data in the corpus share the same definitions and meaning of common nouns and phrases. Thus, words and phrases used within the text will generally have the same meaning, and additional context is not needed to link different instances towards the same count.

LSA is a theoretical and practical method to extract and represent the meanings of words in a text. Using statistical computations, the meaning of words can be estimated from a large corpus of text [42]. Using singular value decomposition, LSA can determine the semantic similarity of words in an attempt to extract meaning [48]. While LSA can begin to extract and represent meaning in a text and identify words and groups that relate. It does so by performing a matrix decomposition over a term-document matrix [49]. Uses of LSA include automated essay scoring [50],

information retrieval, language understanding, metaphor comprehension, and others [51]. One weakness of LSA is in the document-term matrix. As the corpus grows, so does the matrix being evaluated in the algorithm. The larger matrix can cause memory and computational issues.

pLSA is a statistical technique in NLP that defines a proper generative model. Compared to standard LSA which uses a linear transform, pLSA is based on a mixture decomposition derived from a latent class model and yields substantial and consistent improvements over LSA in a number of experiments [43]. Hennig's research provided a proof of concept for evaluating a corpus and summarizing documents based on a pLSA model. They also show that the pLSA model outperforms the LSA model for capturing meaning in a document [52]. The pLSA model provides a statistically reliable model for generating topics when pre-trained over a set of similar documents [53]. Applications of pLSA include image classification [54], machine learning of color names [55], and cross-domain text classification [56]. pLSA is more efficient than LSA in terms of memory and computation by avoiding matrices, but struggles to assign a probability to a new document and may overfit a model due to linear growth of parameters. pLSA also fails to express that a document may express multiple topics, unlike LDA [45].

LDA is a generative probabilistic NLP model that can be used on a text corpus to output results of document modeling, text classification, and topic clustering [44]. LDA generalizes pLSA by changing the fixed document index to a Dirichlet prior thus creating a truly probabilistic generative model [45]. Leveraging LDA as an NLP method can allow for topic identification without the need for whitelisting topics in the course. When implemented correctly, LDA can identify topics and cluster these topics together based on their position in sentences and documents. This mitigates the need to whitelist, thus more efficiently creating TMs and KSATs for courses. LDA

has been applied to topic modeling algorithms in natural language processing, text mining, social media analysis, information retrieval, and many other areas [57].

2.4 Graph Comparison Metrics

We needed to find baseline metrics to compare TMs and KSATs in our research. As discussed in Figure 1 there is a metric in place for comparing TMs that contain similar nodes [7]. This metric will be useful in comparing TMs against a SME-created gold standard. To account for algorithms which may not produce a high number of similar nodes between documents, the rate of topic identification and factors about the SME-created TM should be considered in the analysis.

2.5 Lexical Databases

A lexical database is an organized description that attempts to approximate the lexicon of a language's native speaker. It includes a structure of known morphemes and information about their meanings such as parts of speech designation, a definition, sample sentences to illustrate this sense, cultural annotations to indicate significance, and/or identification of semantic relationships with other morphemes [58].

Princeton's WordNet, a lexical database, contains mainly dictionary-type information, Wiktionary contains additional information such as relation types, languages, a larger volume of terms, instance structure and completeness, and quality of data. These all play vital roles in computing the relatedness of terms and phrases within a corpus of text, and can be much more useful than WordNet alone [59]. Research by Muller shows that Wiktionary can be a useful tool for domain-specific information retrieval [60].

Another lexical database is ConceptNet [61]. ConceptNet contains a far greater amount of data than WordNet, Wiktionary, and several other large data sources.

While the required memory and hardware overhead was higher for ConceptNet than for WordNet and Wiktionary, the additional information ConceptNet uses provides a more robust lexical database. ConceptNet stores a database of relatedness values for words and phrases based on its scraping of web data and an algorithm to score two words from -1.0 to 1.0 based on their relatedness [61]. The ability to quickly query for this value from a local version of ConceptNet negates hardware and build requirements for the database compared to the easier but not as robust WordNet.

2.6 Conclusion

This chapter discussed the tools, techniques, and processes for mapping courses to identify and validate their lesson objectives to assess technical skill. We identified gaps in the research in several domains: asynchronous classroom environments, educational tools, NLP basics, and graph comparison metrics. We also defined the terms used in the rest of this research, including Topic, TM, KSA, and KSAT.

In Chapter III we discuss the methodology for this research, including the tools and algorithms used by PROF, the metrics used to validate TMs and KSATs, and the expected outcomes for each case study.

III. Methodology

3.1 Preamble

In this chapter, we debut the Pedagogical Resource Organization Framework (PROF). PROF consumes unstructured data, identifies topics and their relationships, and creates and visualizes a Topic Map (TM) and Knowledge, Skills, and Abilities Tree (KSAT). Once created, the outputs of the algorithm are tested against a Subject-Matter Expert (SME) created TM to provide a measure of the effectiveness of the algorithms against all courses as a whole and test whether it works better on education or training courses. PROF uses a whitelisting process that requires a custom whitelist for each course evaluated, a Latent-Dirichlet Allocation (LDA) process requiring no customization nor user input, and a modified LDA process that uses ConceptNet [61] to name LDA topics. PROF also creates a KSAT for each case study, which is inspected, and if additional learning paths are identified, a new KSAT is created highlighting these.

We will first discuss methodology of the research and the metrics used to validate the outputs, followed by a discussion of each of the Natural Language Processing (NLP) algorithms supporting PROF.

3.2 Experiment Methodology

The process PROF follows to create TMs and KSATs is split into six steps. Figure 5 details a flowchart of these steps and how they relate to one another. The blue path is required for both, while the black path and black outlined boxes are required only for KSATs. The blue boxes are automated processes. The orange boxes require user intervention in some form. Finally, the green boxes denote a testing process, which requires user intervention and provides results regarding the validity of the

creation process.

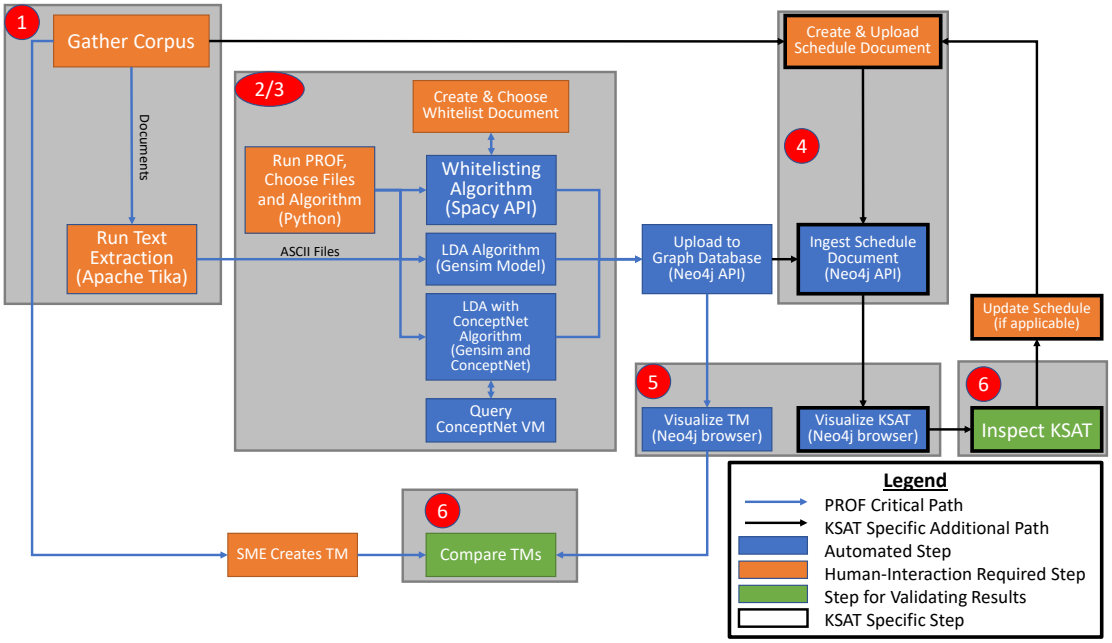

Figure 5: Flow chart for creating, visualizing, and testing TMs and KSATs with PROF

The results of the two comparison metrics can be used to infer which algorithm is best for creating TMs depending on the user's desires. While a SME-Created TM is assumed to be the gold standard for this experiment, three SMEs could create three different TMs for the same course despite having a list of topics to use in the TM. Thus, while the accuracy metric is a percentage, we do not expect it to be greater than 30% in any case.

3.2.1 Limitations

The following limitations should be considered when analyzing and generalizing the PROF methodology.

1. Extracting text from different document types, especially Portable Document

Format (PDF), is challenging. The tool used for this process, Apache Tika [62], provides the best results compared with other available tools [63].

2. To support statistical correctness during text processing, PROF requires a SME to provide a whitelist of topic candidates.

3. To order the course within a KSAT, a schedule document is required to order the lessons, content, and topics present in the course to move from an undirected TM to a directed KSAT.

4. Due to the Neo4j visualization of graph databases, manual placing of nodes in the TMs and KSATs is required. All nodes and relationships are created by the methodology, but their placement on the screen needed to be performed manually.

5. Because of the stochastic nature of LDA, it does not produce the same results, but it does generate consistent topics.

3.3 PROF Algorithm

To create a TM and KSAT with minimal user interaction, PROF follows a multi-step approach. This approach ingests data that constitutes a curriculum and identifies topics and relationships from content items. It then creates relationships to topics based on the chosen algorithm; develops a database with these content items, topics, and links; displays this output in a format navigable for learners; and evaluates output TMs against those manually created by SMEs (Figure 6).

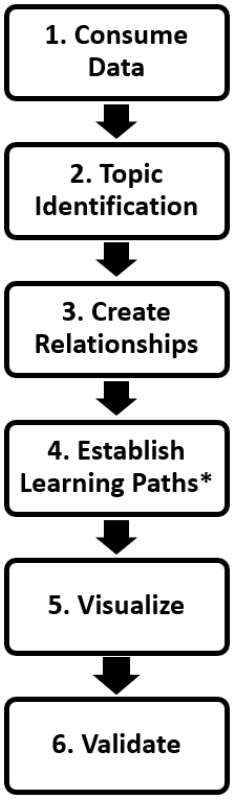

Figure 6: PROF algorithm steps for creating a Topic Map – * denotes step only applicable to a Knowledge, Skills, and Abilities Tree

3.3.1 Consuming Data

Step One of the PROF algorithm extracts text from the corpus and stores it into American Standard Code for Information Interchange (ASCII) [64] files. This section discusses this process.

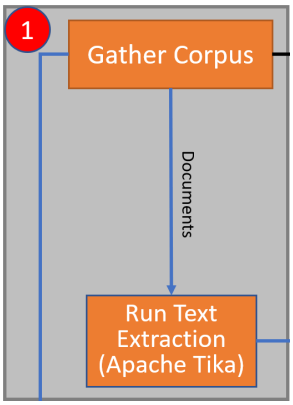

Figure 7: PROF Step 1 Flow Chart

In Step 1, PROF consumes data inputted by the user to be added to the TM. PROF uses Apache Tika [62] to extract text from files, and stores the data in ASCII [64] format for analysis. To extract the text data from the file, we use the command in Appendix A to loop through all PDF files in a chosen directory, extract the plain-text from them, and save them in an adjacent directory as `.txt` files with the same filename as their PDF counterparts.

The next section and its subsections detail the three algorithms PROF uses to identify keywords and relationships within the corpus. Figure 8 details the portion of the overarching PROF flowchart in Figure 5 that is applicable to these algorithms.

3.3.2 Topic Identification

In this section we discuss Step Two of the PROF algorithm and the different techniques used to identify topics and relationships.

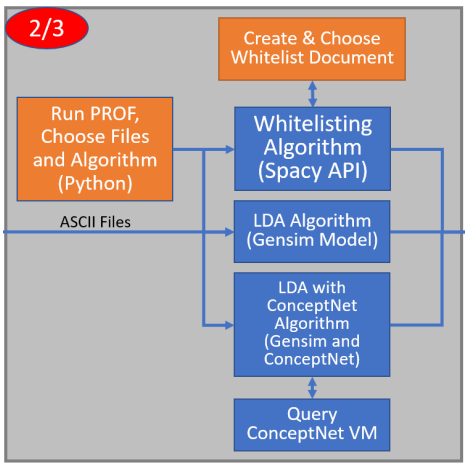

Figure 8: PROF Step 2 Flow Chart

3.3.2.1 Whitelisting

Through a pilot study, the Spacy Application Programming Interface (API) [65] was chosen as the NLP engine of choice to perform the tokenization, lemmatization, and frequency analysis process of the PROF algorithm. This pilot study also aided in decision making about the text extraction tool, Apache Tika [62], which NLP tools to use, as well as the Neo4j [66] graph database [63].

To identify possible topic candidates for inclusion in the TM and KSAT, PROF uses the Spacy API [65] to tokenize and lemmatize the text and find the one-hundred most common noun chunks in the document that are then stripped of preceding stopwords such as "the, a, this, each". The noun chunks then become topic candidates and are stored with their frequency in the document.

A human-generated whitelist of topics is loaded into a data structure to identify the topic candidates that are to be accepted as topic nodes. This whitelist consists of the lesson objectives of the course, pared down to one or multiple nouns or phrases that best describe that objective. These different nouns and chunks are concatenated, separated by slashes, and the incidence of any of the items in the list is counted under

the first of that line.

By formatting the whitelist in this manner, PROF identifies topics in a document that may not otherwise be contextualized. This context issue is noted in initial research with the PROF algorithm, and is due to pronouns, synonyms, or abbreviations used throughout content to refer to a topic that are unrecognizable by a computer [63]. The formatting method in the whitelist helps to address this issue.

After creating a data structure of these whitelisted keywords and their aliases, PROF checks if each of the topic candidates identified by Spacy is present in the whitelist. The strings from the topic candidates data structure that are whitelisted are stored along with their frequency in a newly formed data structure.

Following the creation of a list of accurate topics and relationships, step 3 stores a tuple list of topics that each ingested file contains and the frequency of each topic in the file. The identified topics from each file and their frequency creates a tuple (Figure 9).

$$(file, (topic\_1, frequency), ...(topic\_n, frequency))$$

Figure 9: Tuple format created in Step 3 of PROF Methodology

3.3.2.2 LDA

The second PROF algorithm uses Latent-Dirichlet Allocation (LDA) to identify topics within a corpus of documents, with each topic consisting of a number of words or phrases. LDA was chosen over other topic generation algorithms as it tends to be more accurate than Latent Semantic Analysis (LSA) or Probabilistic Latent Semantic Analysis (pLSA). LDA associates a probability to each word within each topic. These are the probabilities that a topic generated the word. Within Python, the Natural Language Tool Kit (NLTK) and Gensim libraries provided a means for performing

LDA across a corpus. For each document provided, PROF extracts the noun chunks within the documents, then removes the stopwords and lemmatizes the remaining chunks as a way to standardize across the corpus. PROF then creates a data structure containing these strings.

PROF then uses the Gensim [67] corpora to create a dictionary from the tokens and vectorize this dictionary. The LDA model can be created from this matrix. PROF sets the model to query for twenty topics, and runs the model over the corpus twenty times.

Once PROF has generated the model, the algorithm iterates through the generated topics and lists the top five words associated with each topic. These words are stored and used to create the TM.

3.3.2.3 LDA with ConceptNet

In using LDA to identify topics, the naming of LDA-identified topics that matched closely to SME-identified topics was important. A lexical database is used to name topics by leveraging the LDA-generated topics that consist of a set of words and a probability that the topic generates that word. A lexical database can use this information to create a relevant name for that topic.

One of the most difficult pieces of implementing ConceptNet was the creation of a local version that could be queried without limit, as the disk space and RAM needed to decompress the data was very large. The build process required thirteen hours on an Ubuntu 16.04 Xenial virtual machine with 64 GB of RAM, 16 processors, and 300 GB of disk space. Once built, PROF queries the local version using a modified version of a Python class built to interact with the web API [68].

The algorithm takes the LDA-identified topics and finds the terms within the set of five words per topic that are most closely related. These pairs are the name for

topics in the TM being created.

PROF performs queries to find the relatedness of the different topics identified above. These four scores are averaged to find an overall topic relatedness for each topic when compared to each other topic. For example, if Topic One has words A and B and Topic Two has words C and D, the four queries would be comparing A and C, A and D, B and C, and B and D. These four scores would then be averaged to find a relatedness score for Topic One and Topic Two.

3.3.3 Create Relationships

In this section we discuss Step Three of the PROF algorithm.

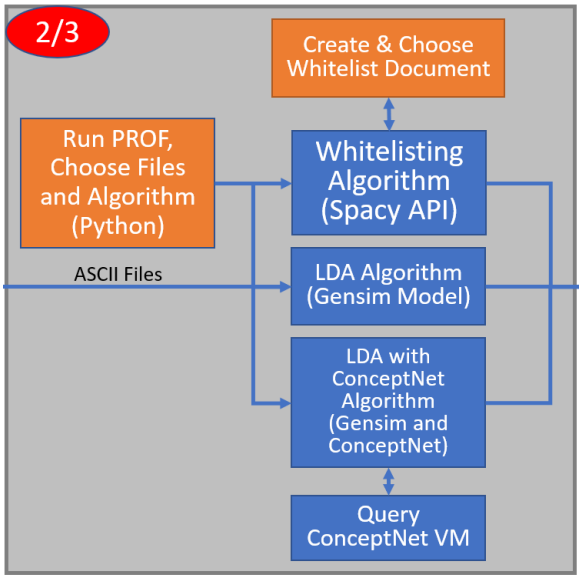

Figure 10: PROF Step 3 Flow Chart

3.3.3.1 Whitelisting Relationship Generation

After storing this data for each file produced by the whitelisting process, PROF loads the content, topics, and their relationships into a graph database. Neo4j uses the Cypher Query Language (CQL) to query and interact with the database.

To load nodes into the database, PROF iterates through the matched topics data structure and queries the database for each topic. A CQL Create statement is written to create a new topic node to link the content node to if the topic is not present in the structure. If the topic is already present, a CQL Create statement is written to link the document node to it.

In the process above, PROF calls a method for checking if a node already exists within the database. This method has parameters of the topic name and the Neo4j driver. It creates a query to match the topic based on its name, and returns the output of a CQL Match query execute command.

PROF calls another method to execute a CQL Match statement. This code takes in the graph driver and a created CQL Match statement, then executes the Match command and tests whether a record object is returned by the database. The method then returns a boolean value for this query.

To connect the topic nodes to one another, PROF queries the database and performs a CQL Merge statement. This statement finds two distinct topics that are referenced by the same content item. Once found, a new relationship is created between the two expressing their co-location in a document. While the relationship is undirected for our purposes, Neo4j cannot create undirected relationships. Therefore, in the CQL statement the created relationship is directed from the topic node with more instances within the corpus as a whole to the topic node with less.

3.3.3.2 LDA Relationship Generation

This algorithm uses the first word of each subtopic identified by the LDA algorithm as the central topic node and creates topic nodes for each of the following four words linked to this central topic node. If a word's node already exists on the TM, this existing node is used, thus creating a TM that is connected between LDA-generated

topics.

3.3.3.3 LDA with ConceptNet Relationship Generation

In the LDA with ConceptNet algorithm, once all topics have a relatedness score against each other, these scores are averaged to create a mean relatedness value between topics. This value is then used to identify relationships between topic names. Each topic pair with a relatedness value above the average is stored and a relationship is created between the topics in the TM.

3.3.4 Establish Learning Paths

In this section we discuss Step Four of the PROF algorithm, which establishes learning paths with a user-generated schedule document to transform a TM into a KSAT.

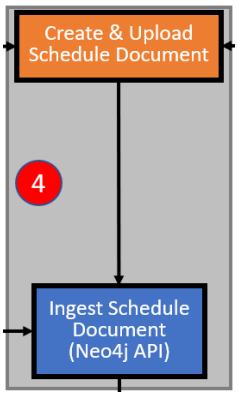

Figure 11: PROF Step 4 Flow Chart

To adapt TMs to KSATs, PROF required more context to create and link lesson nodes to content nodes, direct the graph, and identify learning paths in the course. An additional file is created by the user and input to PROF to provide this context. This file serves as a schedule document that links each lesson or module in the course with the content assigned during that lesson. The file consists of lines with the lesson

or module number comma separated with the filename of content assigned in that lesson (Figure 12).

Lesson Number, Filename of Content Item

Figure 12: Schedule entry format for KSAT Creation in PROF Methodology

With this schedule format, every assigned content item in a course can be mapped to a specific lesson or module. A lesson that assigns multiple content items is represented by multiple lines with that lesson number and the different filenames assigned for that lesson. PROF loops through each line of the schedule file and first creates a node for each lesson in the course, linked together to show the path of the course. For example, Lesson node 1 connects to Lesson node 2, which connects to Lesson node 3, and so on. The code which executes this process is shown in Appendix C.

After creating and linking the lesson nodes, PROF iterates through the document once again to link the lesson nodes to the content nodes. Given that that content nodes are already created and present in the graph database, a match statement is executed to perform this action.

To create links in the KSAT directly from Lesson nodes to Topic nodes, a final CQL statement must be executed. This statement not only creates this connection, but also allows for the KSAT to appear much less cluttered when the database is queried. The CQL query to execute this connection is shown in Appendix D. This displays the KSAT and researchers examine it to identify different learning paths in the course. If different learning paths are identified, a new schedule document can be created that splits these identified learning paths to show how a course could be modularized or shortened to allow students to reach their educational objectives

faster.

3.3.5 Visualize

In this section we discuss Step Five of the PROF algorithm which visualizes the created TM or KSAT.

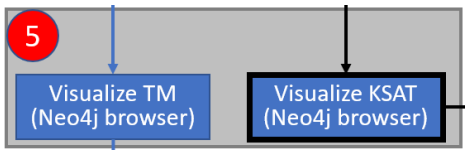

Figure 13: PROF Step 5 Flow Chart

Visualizing the output of the PROF tool requires querying the Neo4j graph database to display the proper subset of the graph to display the TM. To visualize the TM in Neo4j, a simple CQL query to return all topic nodes can be executed. Listing III.1 shows this query.

```
MATCH (t:topic) RETURN t
```

Listing III.1: Neo4j CQL Query to visualize TM

A similar CQL query can be used to visualize a KSAT. Listing III.2 shows this query.

```
MATCH (l:Lesson)-[:ASSIGNS]->(c:content)
MATCH (c:content)-[:references]->(t:topic)
RETURN l,t
```

Listing III.2: Neo4j CQL Query to visualize a KSAT

3.3.6 Validation

In this section we discuss Step Six of the PROF algorithm as it pertains to TMs, including how the TMs are evaluated.

3.3.6.1 Topic Map Comparison

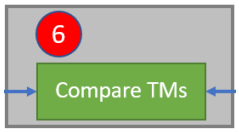

Figure 14: PROF Step 6 Flow Chart for TMs

Following the creation of TMs with the different algorithms, TMs are compared against SME-Created TMs to assess their accuracy. Measures of success for the NLP method is topic recognition rate by PROF compared to topics present in the SME-Created TMs. This rate provides a general success level for the NLP and provide insight into whether future methods should increase the number of topic candidates per document in the whitelisting, more topics per corpus with LDA, or use a different NLP method if the topic identification rate is lower than expected. The topic recognition rate measures how many topics present in the SME-Created TM are identified by the PROF algorithm.

We will use the McClure method to compare PROF-Created TMs against the SME-Created TMs. This metric provides a numerical value in terms of similarity between the TMs. By providing PROF with a user-generated whitelist of all the topics present in the SME-Created TM, and performing the calculation across all similar nodes, the different PROF-Created TMs can be compared individually against the SME-Created TMs. The SME-Created TMs serve as a baseline for initial comparisons, and the changes in accuracy between different TMs provide insight into the merit of the NLP process used for a TM. By considering both the topic identification rate and the McClure similarity score, we can better assess the overall results of the different algorithms used and how each compares to the SME-Created TM.

3.3.6.2 KSAT Comparison

In this section we discuss Step Six of the PROF algorithm as it pertains to KSATs, specifically in identifying the possibility of alternate learning paths based on the clustering of topics and updating the schedule document to demonstrate these paths.

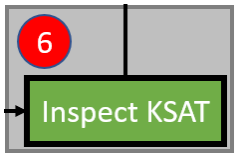

Figure 15: PROF Step 6 Flow Chart for KSATs

The courses used for the experiment are synchronous. Therefore SME-Created KSATs were not needed for comparison. Instead, the schedule document ingested by PROF creates a KSAT for the course in its current form. This KSAT can then be examined to identify different groupings of topic nodes that relate to one another. Topic groupings are different learning paths within the course. The distinction as to whether or not one grouping requires another as a prerequisite relies on the user having some background knowledge. However, the visualization of these groups can provide insight into how a course is structured and lower the need for knowledge to initially identify and separate these paths.

To demonstrate how to identify shorter learning paths, the case studies in this research are mapped into TMs and KSATs using the whitelisting algorithm. The KSATs will be inspected, and the schedule document will be altered and retested to implement multiple learning paths based on the clusters found within the original single-track KSAT. Students can follow these different paths to target specific outcomes at a potentially faster rate than by completing the entire course.

3.4 Conclusion

This chapter details the research methodology, the PROF algorithm, and the different tools and techniques PROF leverages in each step. We also propose a methodology for validating the TMs and KSATs PROF produces.

Next chapter, we examine the output TMs for each of the courses and algorithms, the original and, if applicable, modified KSATs created for each course. We will also compare the TMs to the SME-Created TMs and discuss the case studies' performance.

IV. Results and Analysis

4.1 Preamble

This research consists of creating a Topic Map (TM) and Knowledge, Skills, and Abilities Tree (KSAT) for each course using the Pedagogical Resource Organization Framework (PROF) whitelisting, Latent-Dirichlet Allocation (LDA), and LDA with ConceptNet naming algorithms. We then compare the TMs to those created by a Subject-Matter Expert (SME) to judge their accuracy. We will also compare the values for TMs for the same case study with different algorithms to find initial conclusions about which algorithm is more efficient. Four case studies were included in the experiment, including two education courses hosted at the Air Force Institute of Technology (AFIT), and two Air Force training courses. The first two courses tested are the Computer Science and Computer Engineering (CSCE) 660 course and the Network Warfare Bridge Course (NWBC). The pilot study tested these courses and the results guided the initial research [63]. This research includes those two courses. The other two courses are the Systems Engineering (SENG) 593 and Model-Based Systems Engineering (MBSE). We compare the results of these courses against one another to attempt to find differences in the effectiveness of the methodologies on training courses versus education courses. Each TM will be evaluated and scored against the SME-Created TM, and the KSATs will be altered to show different learning paths that may be present in the course. The output of the experiment will be three TMs and scores for each course, and a baseline KSAT, and an altered KSAT, if applicable. The research found that the topic identification rate and McClure score was highest across all case studies when using the whitelisting algorithm for all but one instance where the McClure score for the LDA TM was higher. Additionally, the LDA algorithm with ConceptNet performed the poorest in these two metrics for

all case studies. These findings suggest that as human interaction in TM creation decreases, so too does topic identification and accuracy of the TM. We also found additional learning paths in two of the case studies. These learning paths seem to be viable methods for modularizing course content.

Next section, we discuss the limitations of the results of this research. Then, the resultant TMs of the research, split by case study. After all case studies have been presented and interpreted individually, we will analyze the results as a whole.

4.2 Limitations

The following limitations should be considered when analyzing and generalizing the results of the experiment.

1. With SME-generated TMs, there is an inherent bias introduced, and a single SME may present a solution in a set of possible solutions. This experiment only tests the algorithm's ability to match the SME's opinion and not necessarily the algorithm's ability to produce an effective TM.

2. For the CSCE 660 case study, pruning of the SME-generated TM was required to remove topic nodes not present in the documentation, as the documentation had changed since the SME-generated TM was created. This is the only instance of pruning in the experiment but may skew the results for that case study.

3. The McClure similarity metric for comparing TMs does not directly penalize a TM for possessing more nodes and relationships than the SME-generated TM. Thus TMs with more nodes and relationships are more likely to have a similarity score using this metric.

4.3 Case Studies

The next four subsections will present the results of each case study and discuss the results for topic identification rate and McClure score for each TM.

4.3.1 CSCE 660

The first case study tested with the PROF algorithm was CSCE 660 Mobile and SCADA Security Course. This course is a graduate-level course at the Air Force Institute of Technology [1][63]. A SME created a TM for this course in previous research, and the topics in the SME-Created map were used as a baseline for topic recognition measurement.

The SME-Created TM for the course was not based directly off of assigned course readings and material. It was created from relevant topics within and adjacent to the course which may not be explicitly stated in readings or may be tangential to the course, not all of the nodes in the SME-Created TM were present in course text. Of the 121 topics included in the SME-Created TM, only 74 were directly mentioned in the text of the course readings. Because the PROF algorithm and NLP could not account for these differences, the SME-Created TM was adjusted to remove nodes not present in the course documents, and their relationships were bridged across the missing nodes. For example, if Topic A was connected to Topic B, and Topic B was connected to Topic C, but Topic B was not present in the course readings, Topic B was removed, and Topic A was connected to Topic C. The adjusted TM is shown in Figure 16.

[1]Dr. Tim Lacey was the professor of this course for the version the Topic Map modeled

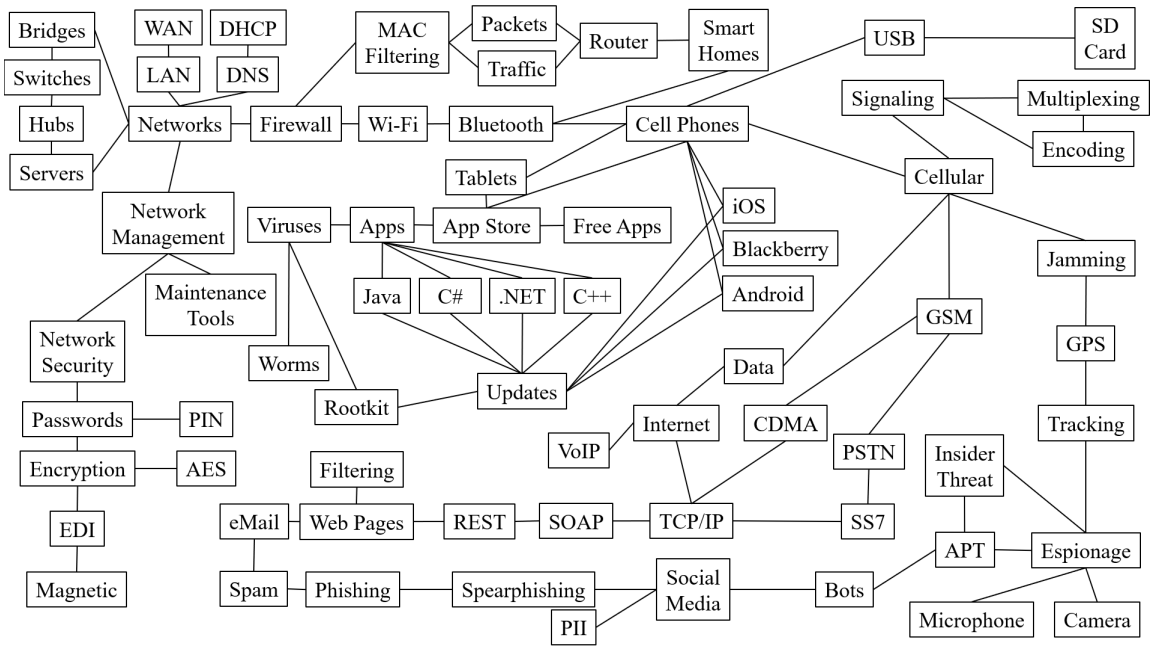

Figure 16: Adjusted SME-Created TM for CSCE 660 Course

4.3.1.1 Whitelisting

After applying the PROF-whitelisting algorithm on the CSCE 660 Course content, it identified 46% (34 of 74) topics found in the SME-Created TM. This data was then used to connect topic nodes based on their relationships within the course text. Figure 17 is the TM created by the PROF whitelisting algorithm.

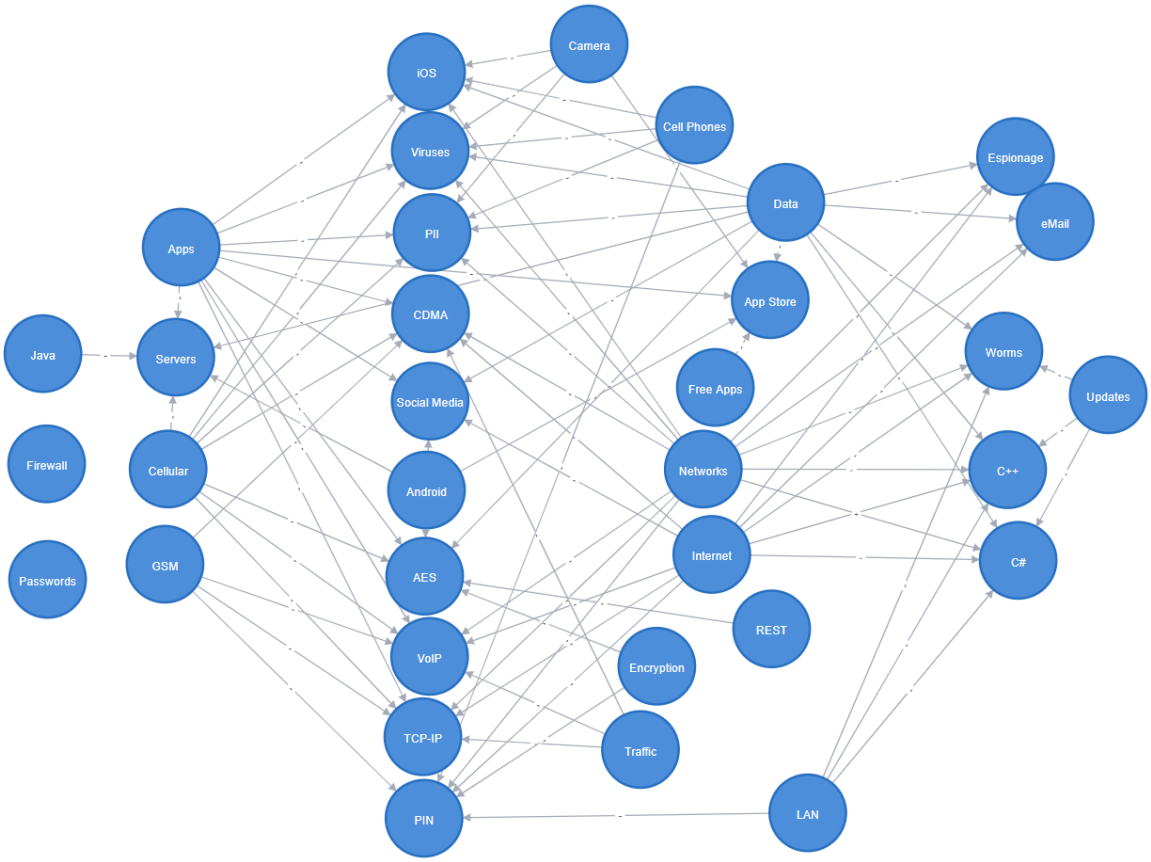

Figure 17: PROF Whitelisting Topic Map for CSCE 660 Course

4.3.1.2 LDA

Figure 18 is the PROF-LDA algorithm TM created for the CSCE 660 course.

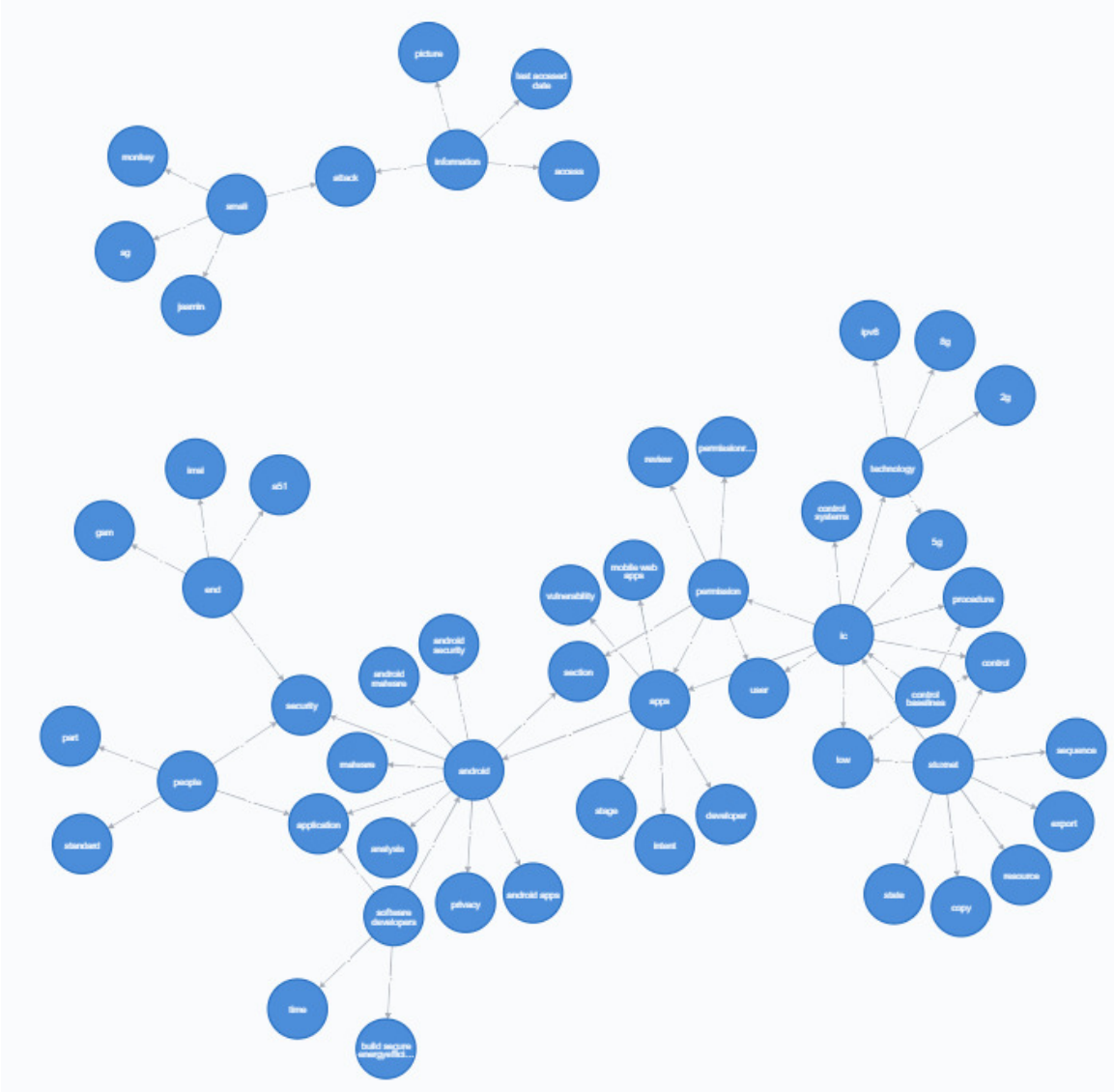

Figure 18: PROF LDA-algorithm TM for CSCE 660

4.3.1.3 LDA with ConceptNet

Figure 19 is the TM the PROF-LDA with ConceptNet algorithm created for the CSCE 660 course.

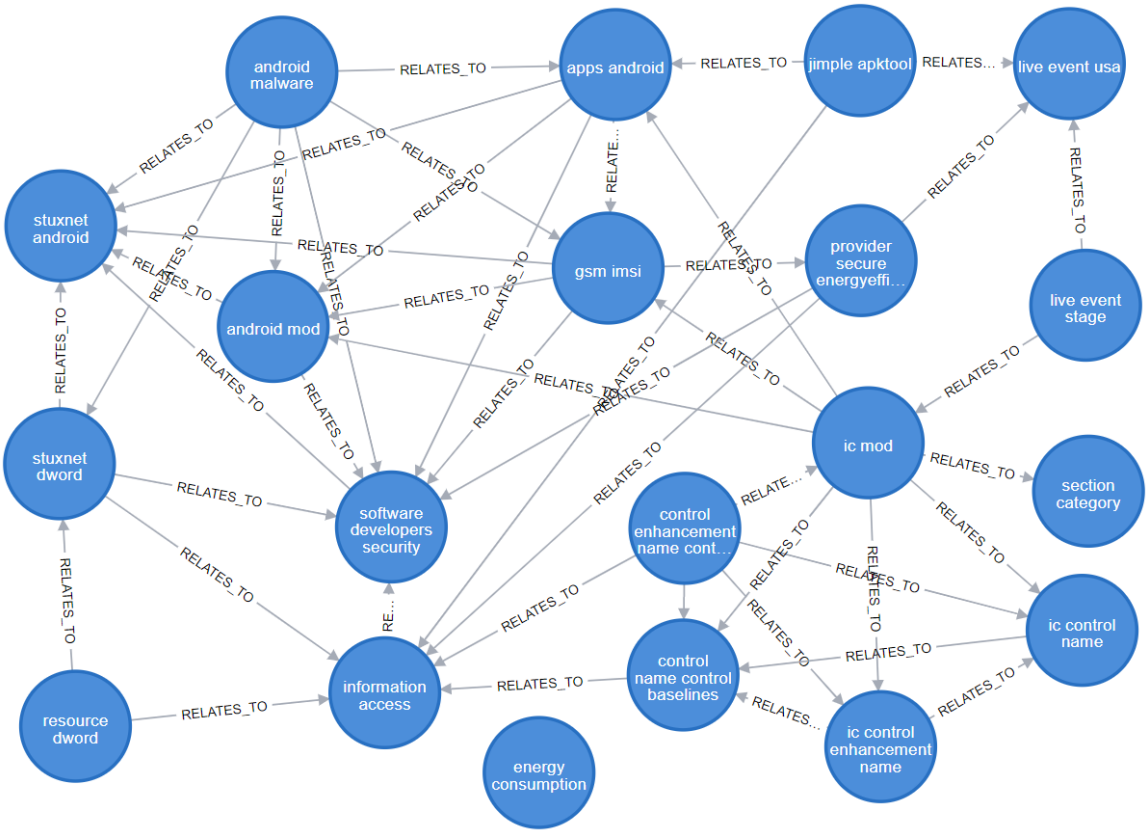

Figure 19: PROF LDA-algorithm with ConceptNet naming TM for the CSCE 660 Course

For this case study, the SME-created TM was created for a previous version and needed to be amended to ensure it only contained topics mentioned within the course's text. While this gave every algorithm an opportunity to identify 100% of topics, the lower than expected level of topic identification is due to the low number of topic references in the course text. As most files in this case study were academic papers, there was not an emphasis on repeating or emphasizing topics throughout the course like in a textbook that has multiple chapters read throughout the length of the course; this causes a lower topic identification rate for the algorithms. The amending of the SME-created TM also may have caused issues by not containing some relationships, thus impacting the McClure score.

4.3.1.4 KSAT

Figure 20 is the initial KSAT created for the CSCE 660 Course.

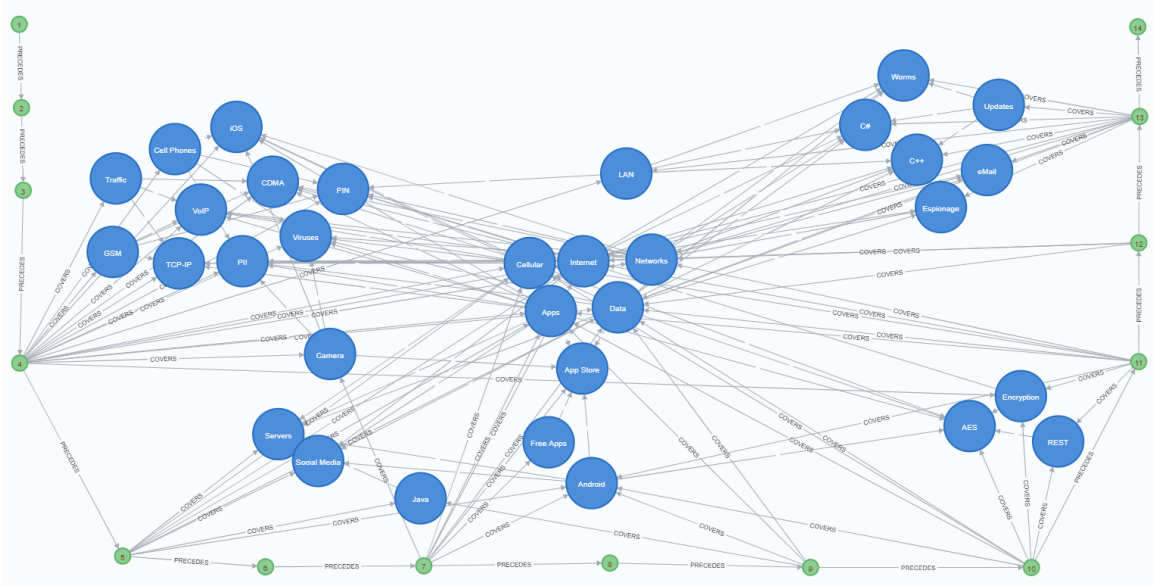

Figure 20: PROF Whitelisting KSAT for CSCE 660

The KSAT was visually examined, and three different clusters of topics were identified. These are the topics linked to lessons one through four, on the left hand side of the KSAT, the topics linked to lessons five through nine, on the bottom of the KSAT, and finally those linked to lessons ten through fourteen, on the right side of the KSAT. The updated KSAT (Figure 21) for the course with these three learning paths split, all beginning from lesson one and moving outwards from the central lesson node.

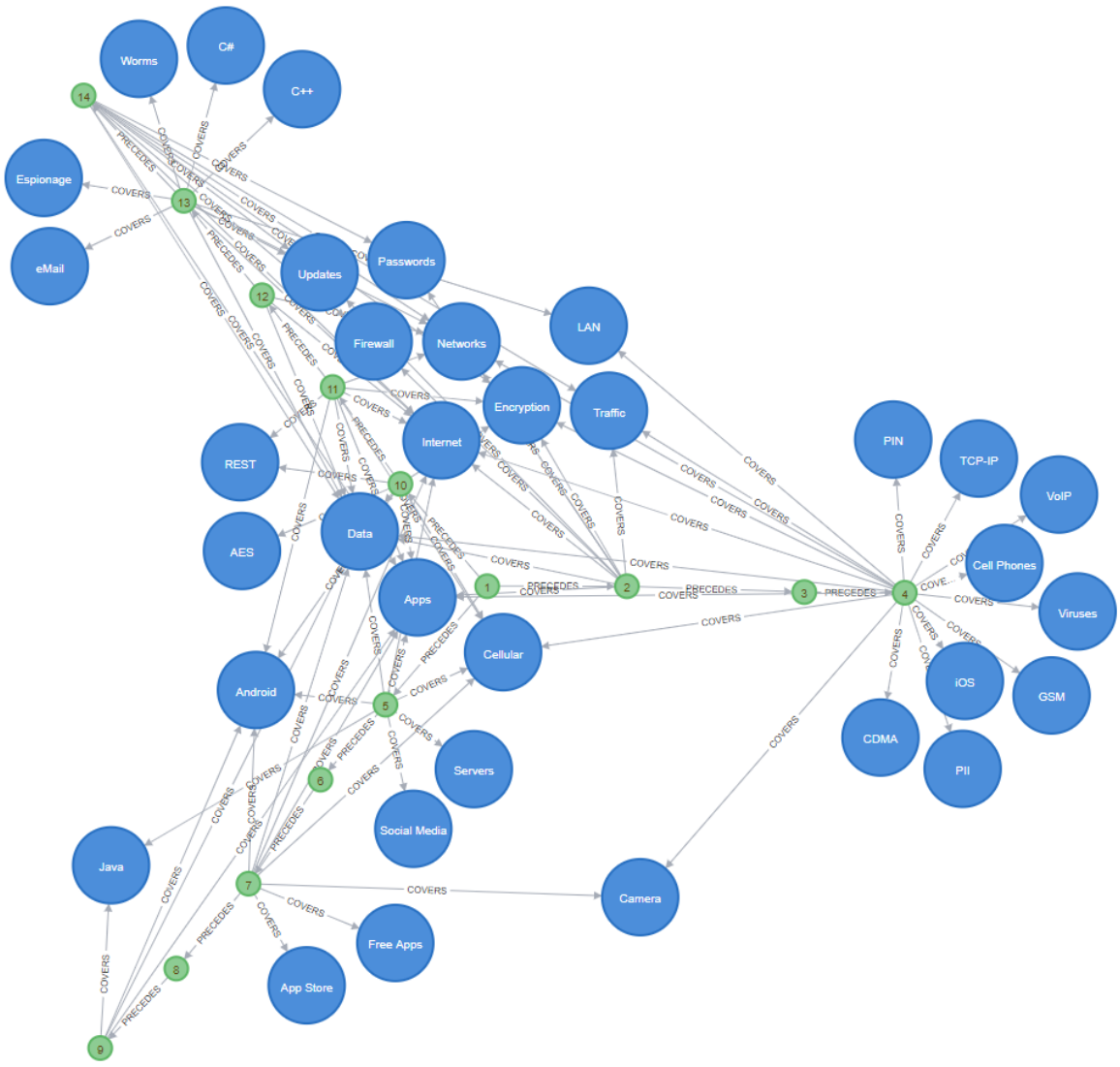

Figure 21: Updated KSAT for CSCE 660 with 3 learning paths

4.3.2 SENG 593

The SENG 593 Agile Software Systems Engineering Course is a graduate-level course hosted at the Air Force Institute of Technology. The SENG 593 Course consists of 84 files, including HTML, PDF, Microsoft Word, and Microsoft PowerPoint file types. A SME[2] created a TM for the course, shown in Figure 22.

[2]Dr. Mark Reith, a curator and designer of the course, is the SME referenced here.

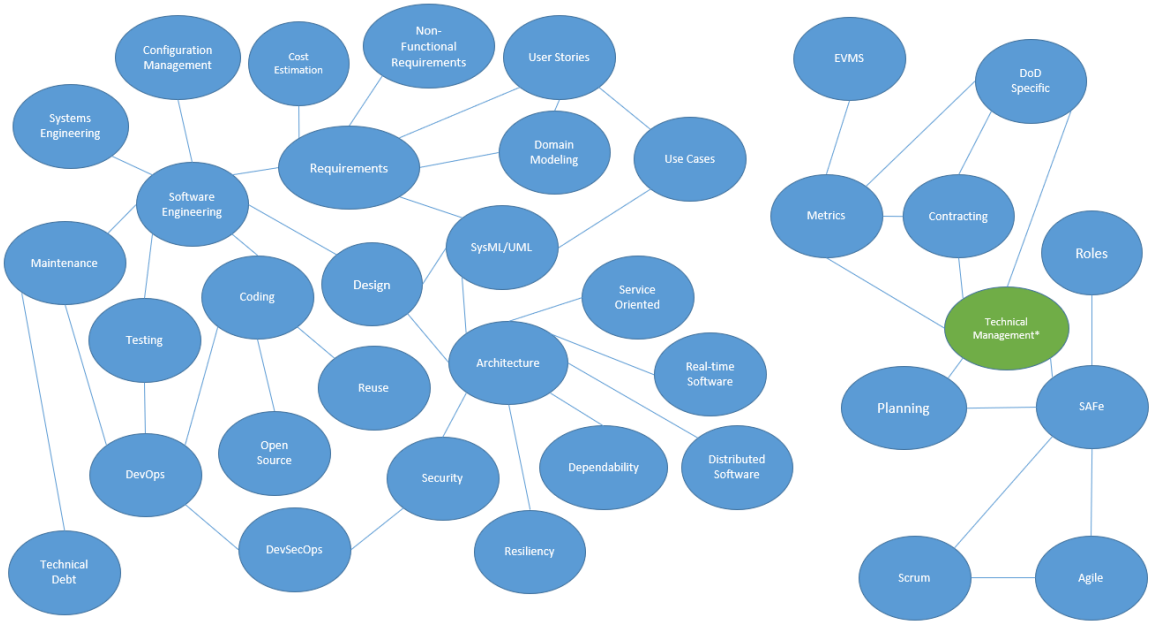

Figure 22: SME-Created TM for SENG 593

4.3.2.1 Whitelisting

The PROF whitelisting algorithm identified 81% (29 of 36) of the topic nodes found in the SME-Created TM. PROF also output a TM for the course, shown in Figure 23.

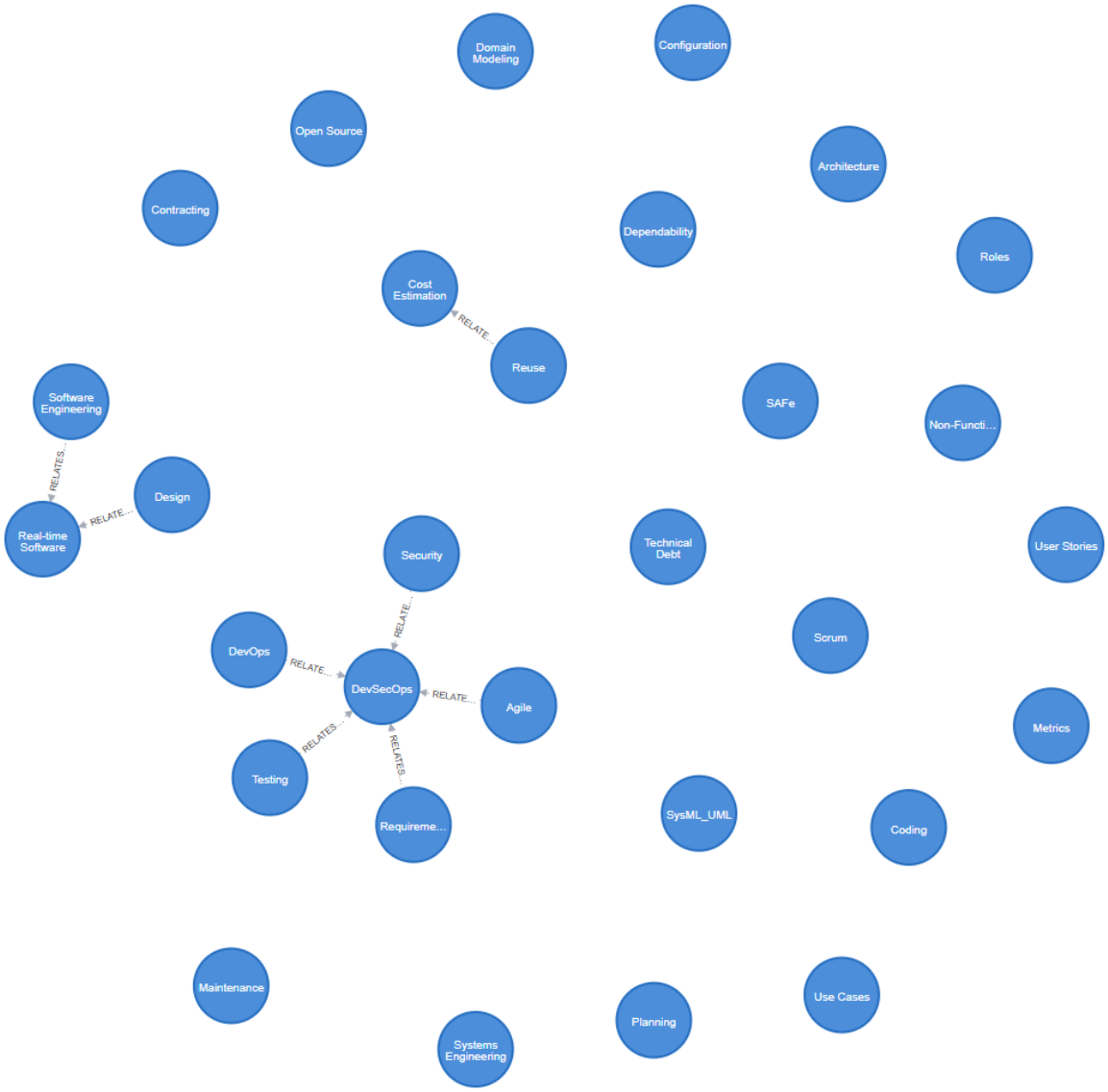

Figure 23: PROF Whitelisting TM for SENG 593

The SENG 593 Course TM (Figure 23) contains less than one connection per node, less than those shown in Figure 17, Figure 27, and Figure 33. We hypothesize that this is due to a much larger number of files being present in the course material. The type of material included in this course includes academic articles and textbook excerpts and course slides which normally include a lecture. Including lecture with slides makes it difficult for PROF to recognize the different topics being emphasized

in the presentation because the main topic is semantically similar to the tangential topics in the lesson. An additional issue is the query PROF uses to link topics together. This query is meant to be non-greedy in that it queries for topics with only one document referencing them first in an attempt to prevent an overload of connections in the TM. Because the documentation of the SENG 593 course contains so many references to the topics throughout the content items, the query does not perform as well in identfying and creating relationships as it does with the other case studies. These other case studies are much more segregated in terms of each content item invoking a small set of topics without much overlap between course readings.

4.3.2.2 Latent-Dirichlet Allocation

Figure 24 illustrates the TM generated for the SENG 593 course, the other educational course case study, by the PROF LDA algorithm.

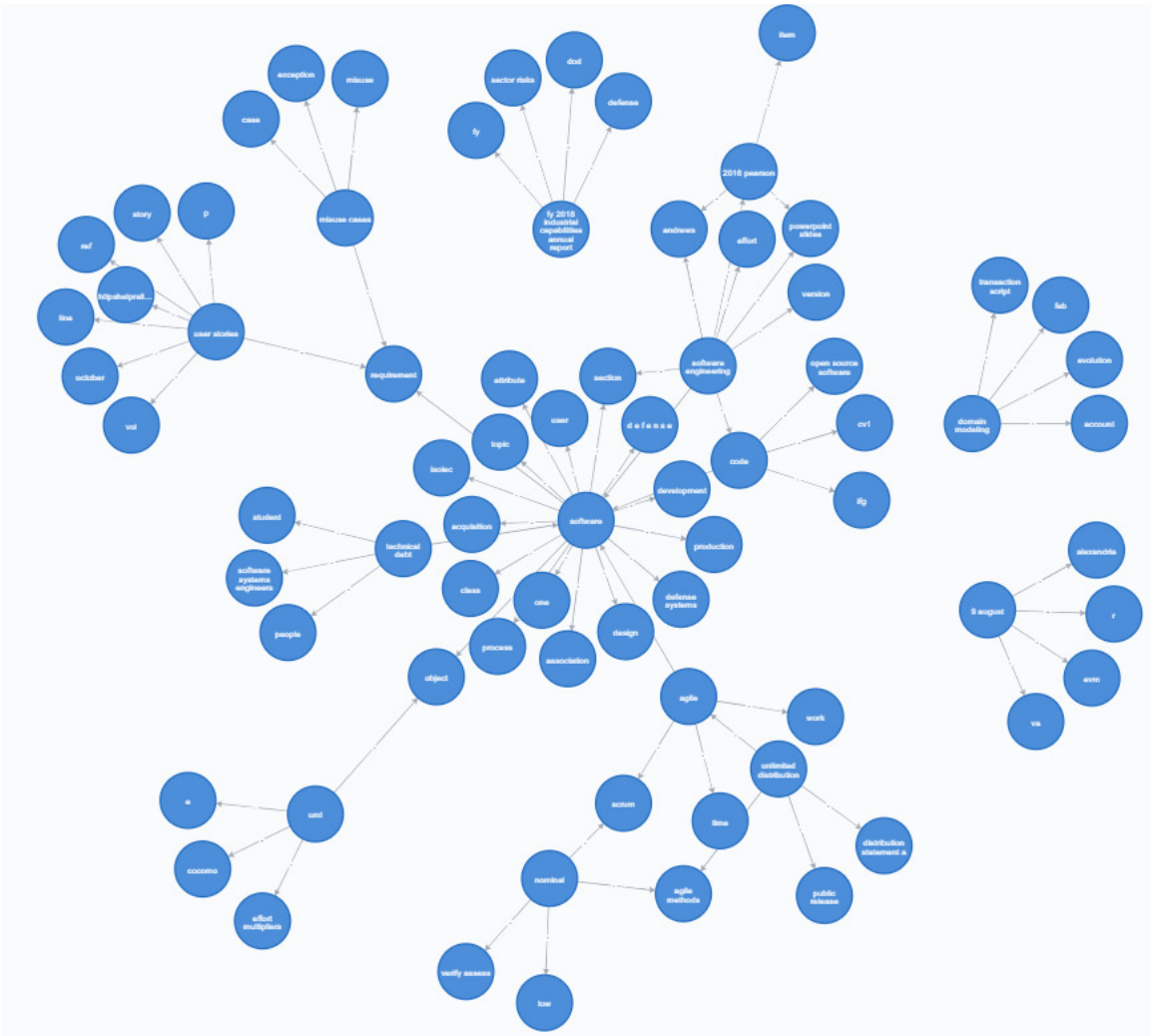

Figure 24: PROF LDA-algorithm TM for SENG 593

4.3.2.3 LDA with ConceptNet

Figure 25 details the TM created by the PROF LDA algorithm with ConceptNet for the SENG 593 course.

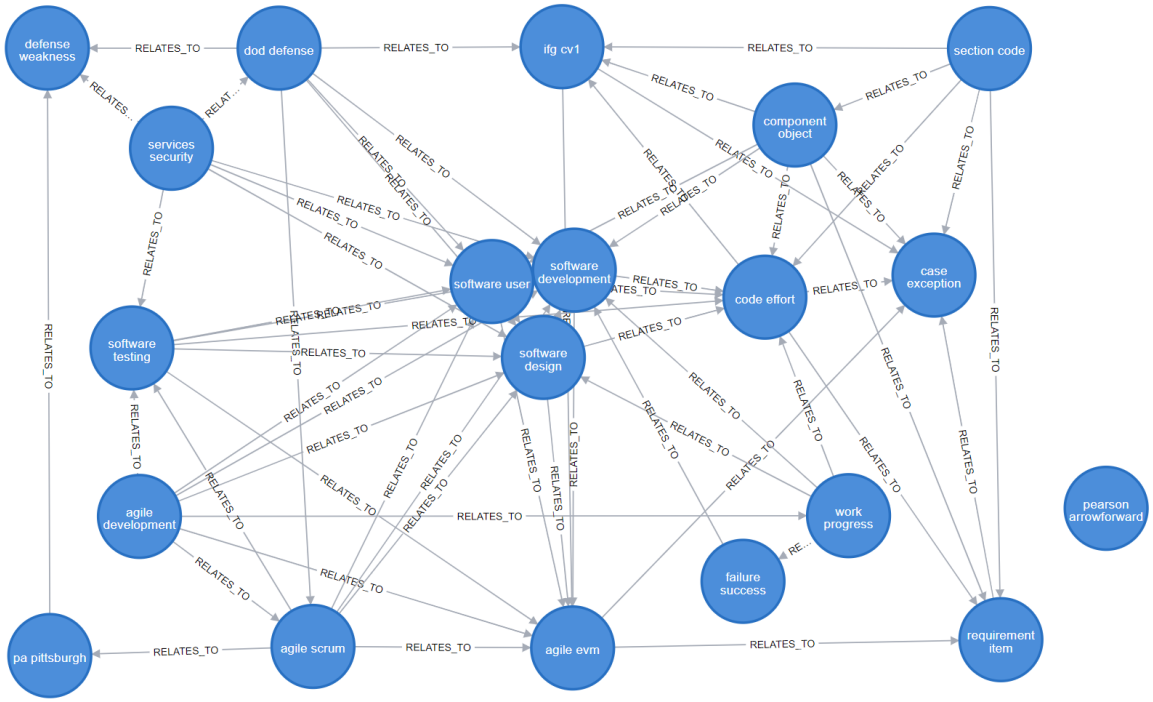

Figure 25: PROF LDA-algorithm with ConceptNet naming TM for the SENG 593 Course

The SENG 503 case study is the second educational course and has different results than the CSCE 660 case. This may be due to the nature of the SME-Created TM; it was created for this iteration of the course, contains concise topics, and a text corpus that is large with direct references to topics. These reasons are cause for a high rate of topic identification not only in the whitelisting algorithm, but the LDA and LDA with ConceptNet naming algorithms as well. The latter two algorithms may have performed well for this course because the SME-identified topics consisted of single words or short phrases, making them more identifiable in the text for the Natural Language Processing (NLP) tools and fitting for the LDA algorithm.This makes LDA more likely to match the SME topics in both generic and ConceptNet naming compared to more specific listing. As for the McClure score, the reasons above for the topic identification success can be attributed to an increased score, as well as

the layout of the SME-Created TM, which had more direct relationships between key course concepts than other SME-Created TMs.

4.3.2.4 KSAT

Figure 26 shows the KSAT for created for the SENG 593 course.

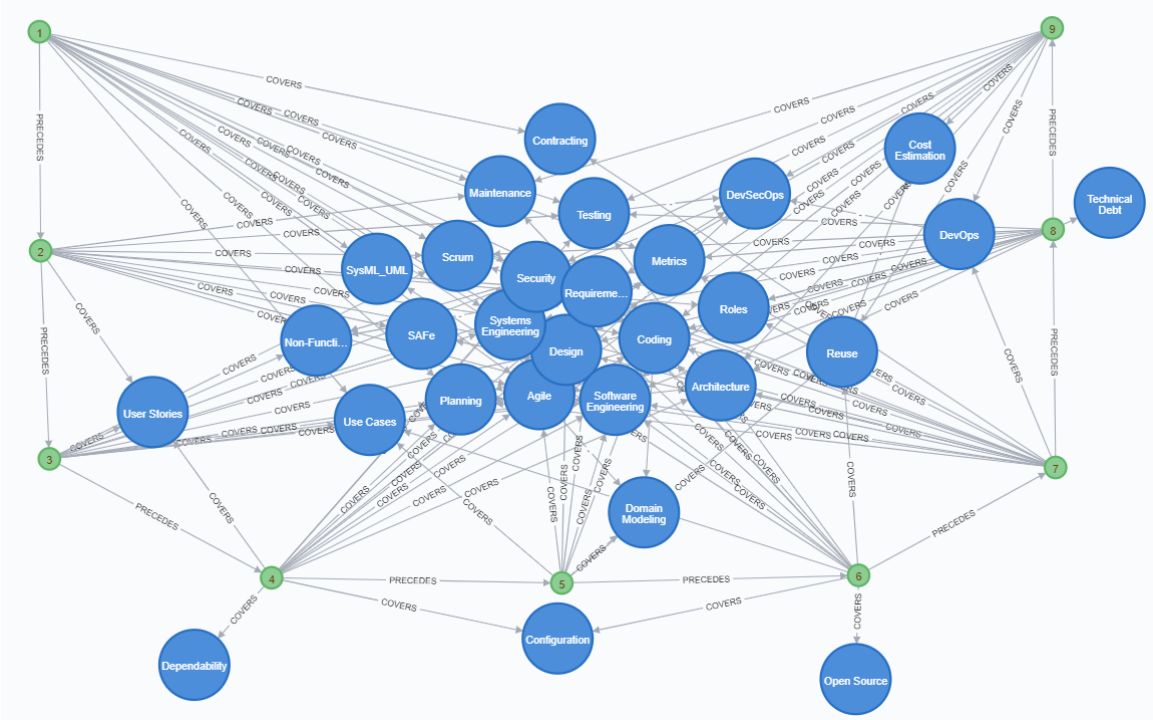

Figure 26: PROF Whitelisting KSAT for SENG 593

The reasons cited above demonstrate why the SENG 593 case study may have performed so well in the topic identification rates and McClure score compared to the other case studies. These characteristics, topics that are present throughout course documents and a fewer number of lessons (seven compared to fourteen for CSCE 660 and 20 for NWBC), led to no obvious distinct clusters of nodes for the SENG 593 course. Because no clusters were found, an updated KSAT was not created.

4.3.3 NWBC

The first training course case study is the NWBC. This training course consisted of numerous study guides, course briefings, exercises, and readings. PROF's whitelisting algorithm identified 51% (68 of 133) of topics in the SME-Created TM for the NWBC.

4.3.3.1 Whitelisting

PROF used this data to produce a TM for the course (Figure 27).

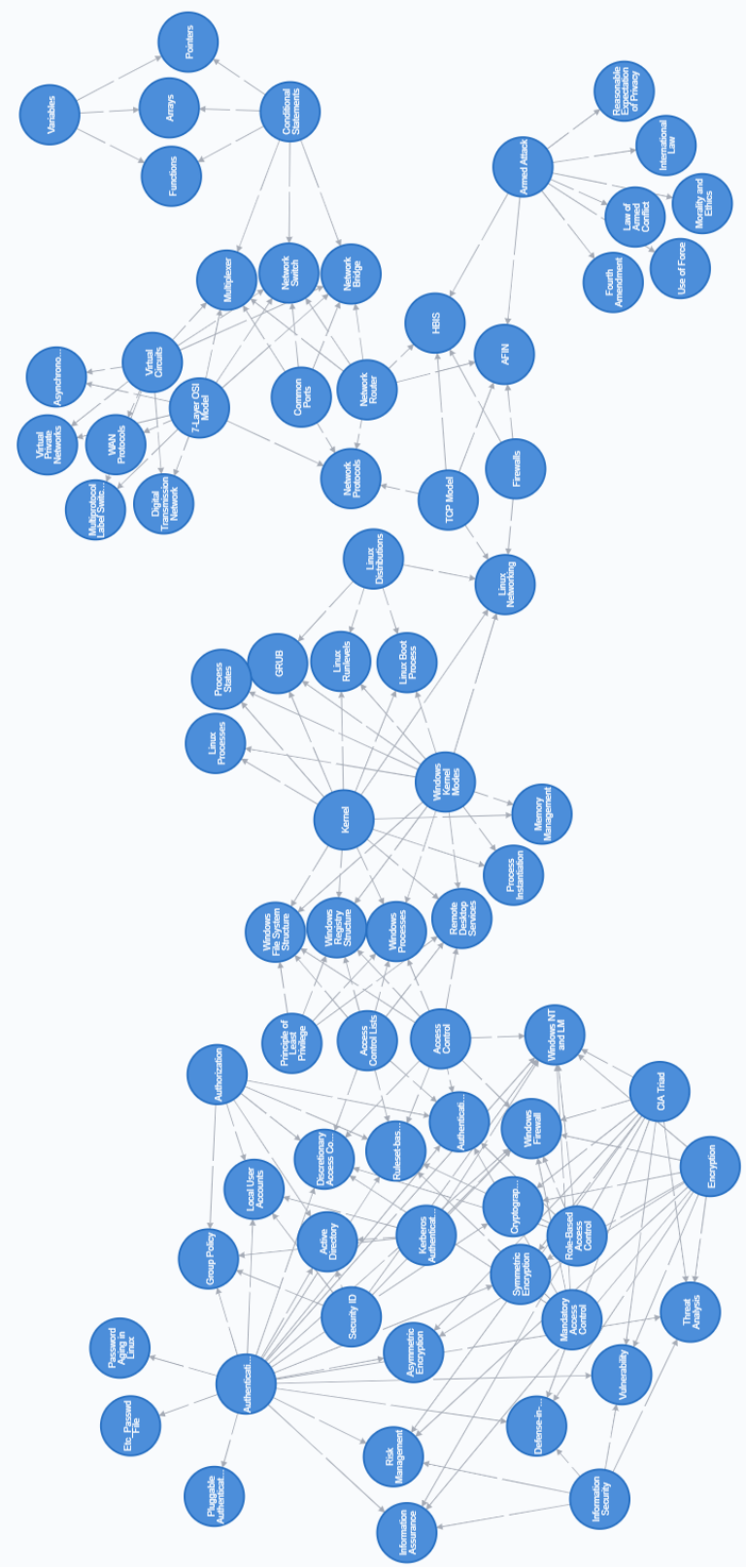

Figure 27: PROF Whitelisting Topic Map for NWBC

4.3.3.2 LDA

Figure 28 is the TM created by the PROF LDA algorithm for the NWBC.

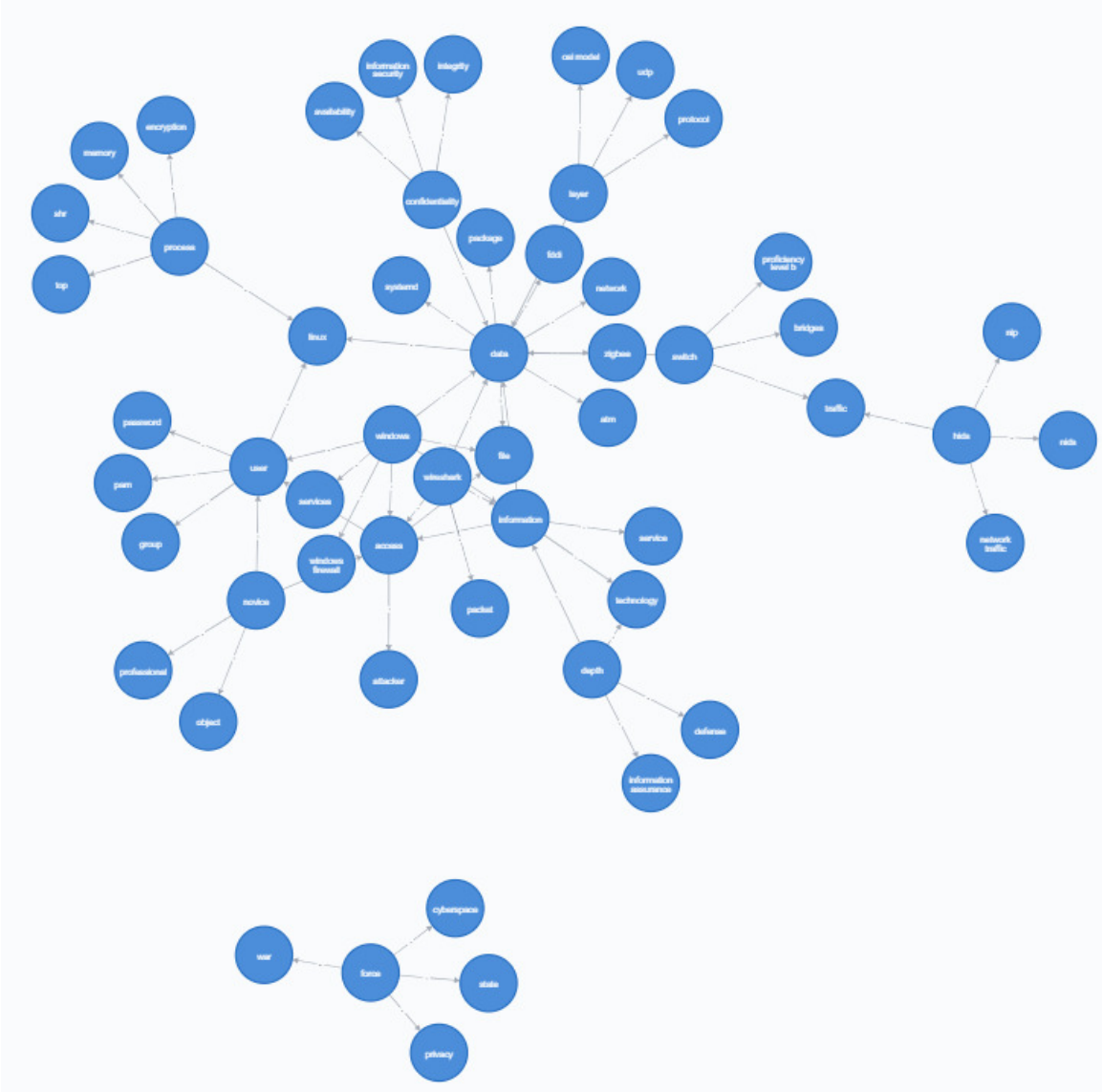

Figure 28: PROF LDA-algorithm TM for the NWBC

4.3.3.3 LDA with ConceptNet

Figure 29 is the TM created for the NWBC using the PROF LDA with ConceptNet algorithm.

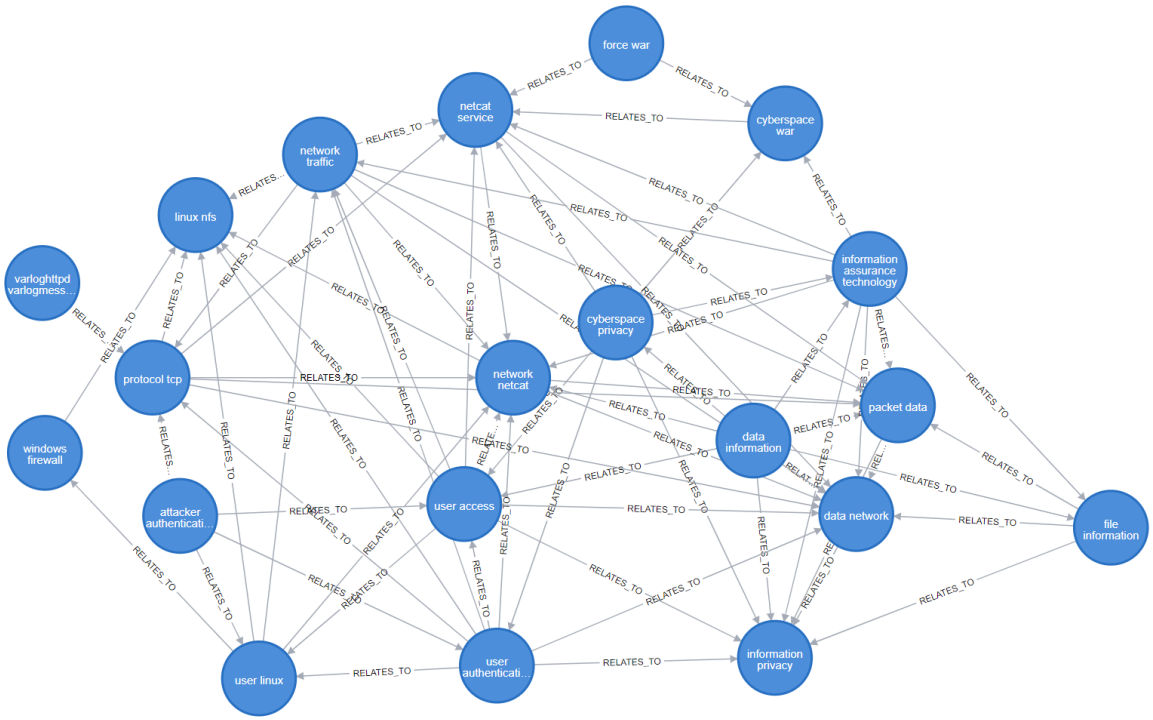

Figure 29: PROF LDA-algorithm with ConceptNet naming TM for the NWBC

The first of two training course case studies, the NWBC produced higher results in both topic identification rate and McClure score compared to the MBSE. The topic identification rate is most likely due to the training documents provided in the corpus for the course. These documents are essentially study guides, iterating lesson objectives at the beginning and emphasizing key nouns and phrases, SME-identified topics, throughout their body. This leads to a higher identification rate. However, the McClure score for this course is lower than both education courses for the whitelisting algorithm. This is due to a large number (greater than one-hundred) of topics in the SME-Created TM for this course. This high number of topics may lead to a lower likelihood that two topics will have a relationship, thus driving down the McClure score. While it has a lower score in the whitelisting algorithm, however, it retains its score throughout the three rather well. This is because the topics and relationships identified by the whitelisting algorithm are more concisely named and more prevalent

in course documents, giving LDA and ConceptNet a higher likelihood of identifying them and PROF a higher likelihood of creating those relationships.

4.3.3.4 KSAT

Figure 30 is the KSAT for the NWBC.

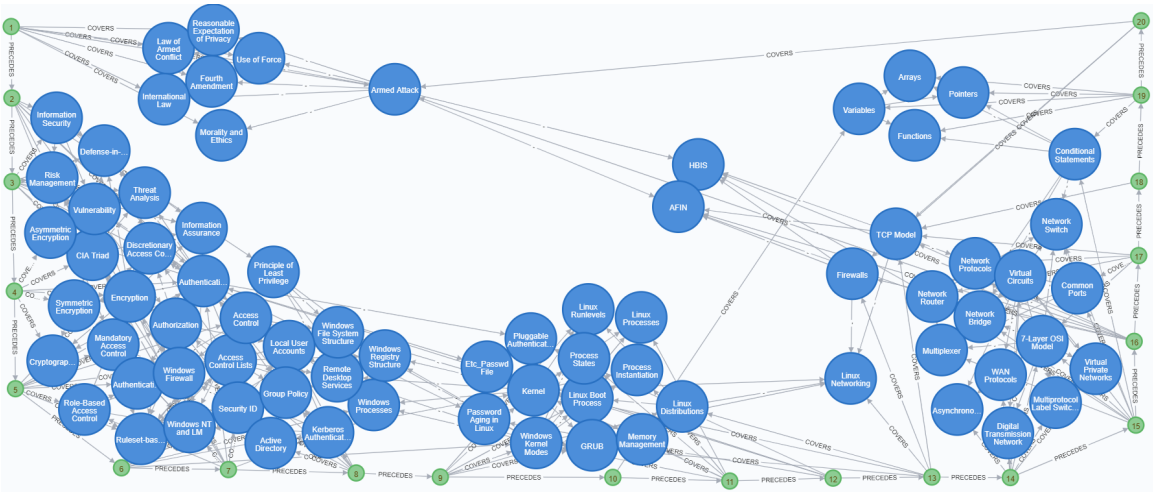

Figure 30: PROF Whitelisting KSAT for NWBC

In the NWBC, even more learning paths were identified. These five paths consist of the first lesson as a standalone, lessons two through eight, in the left and bottom left of the KSAT, lessons nine through thirteen, in the bottom middle of the KSAT, lessons fourteen through eighteen, in the bottom right and right of the KSAT, and finally lesson nineteen as a standalone. Lesson twenty does not link directly to any topics in the KSAT and is not considered its own learning path. Figure 31 is the updated KSAT for this course, with each learning path moving away from the lesson one node in the center.

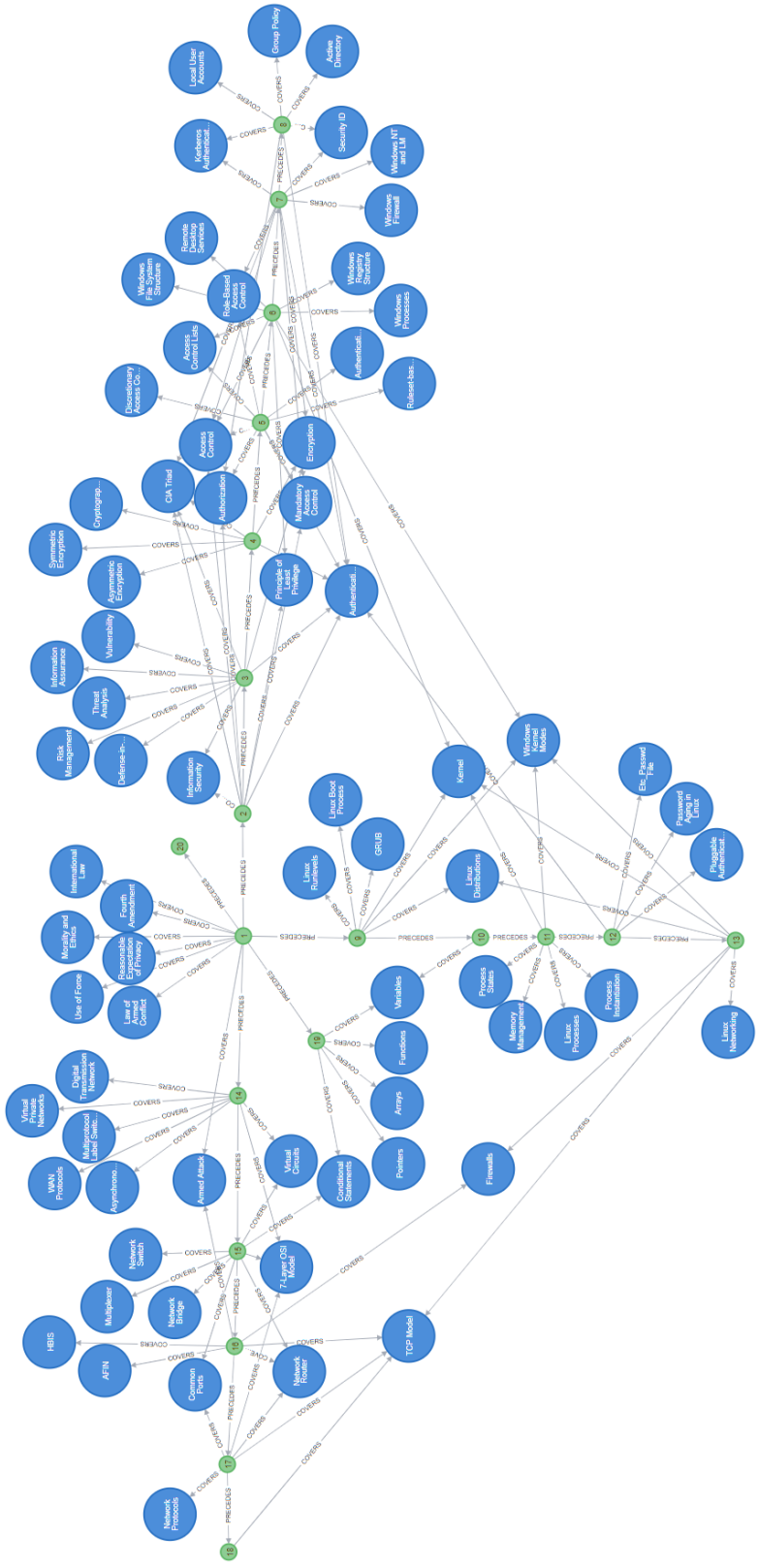

Figure 31: Updated KSAT for the NWBC with 5 learning paths

4.3.4 MBSE Course

The Model-Based Systems Engineering (MBSE) Course, a technical short course hosted at the Air Force Institute of Technology, is the final case study. Figure 32 shows the SME-Created TM, created by Dr. Thomas Ford, for the MBSE Course.

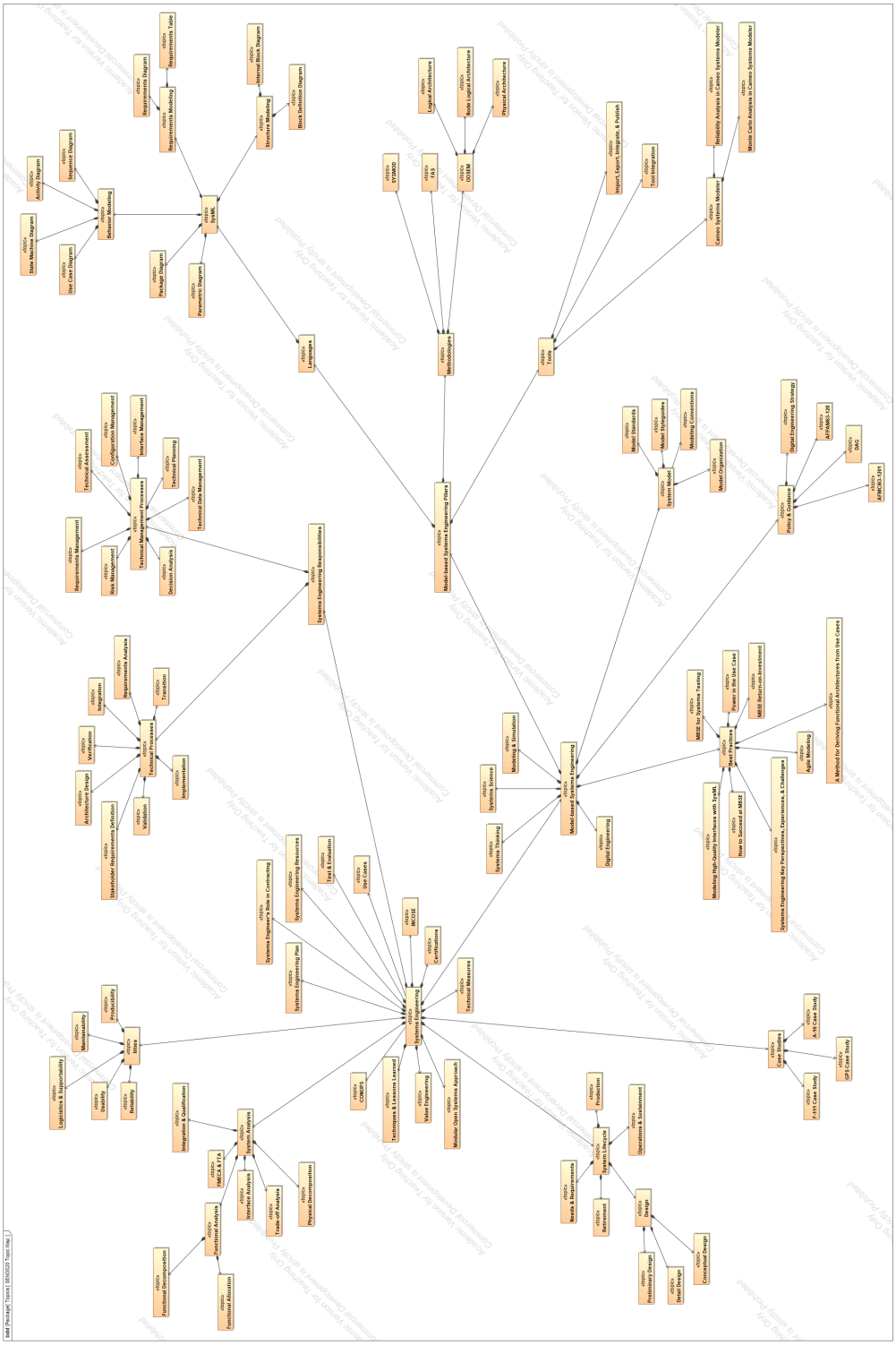

Figure 32: SME-Created TM for MBSE Course

4.3.4.1 Whitelisting

The PROF whitelisting algorithm processed the data files used to teach the MBSE course and produced a TM, as shown in Figure 33.

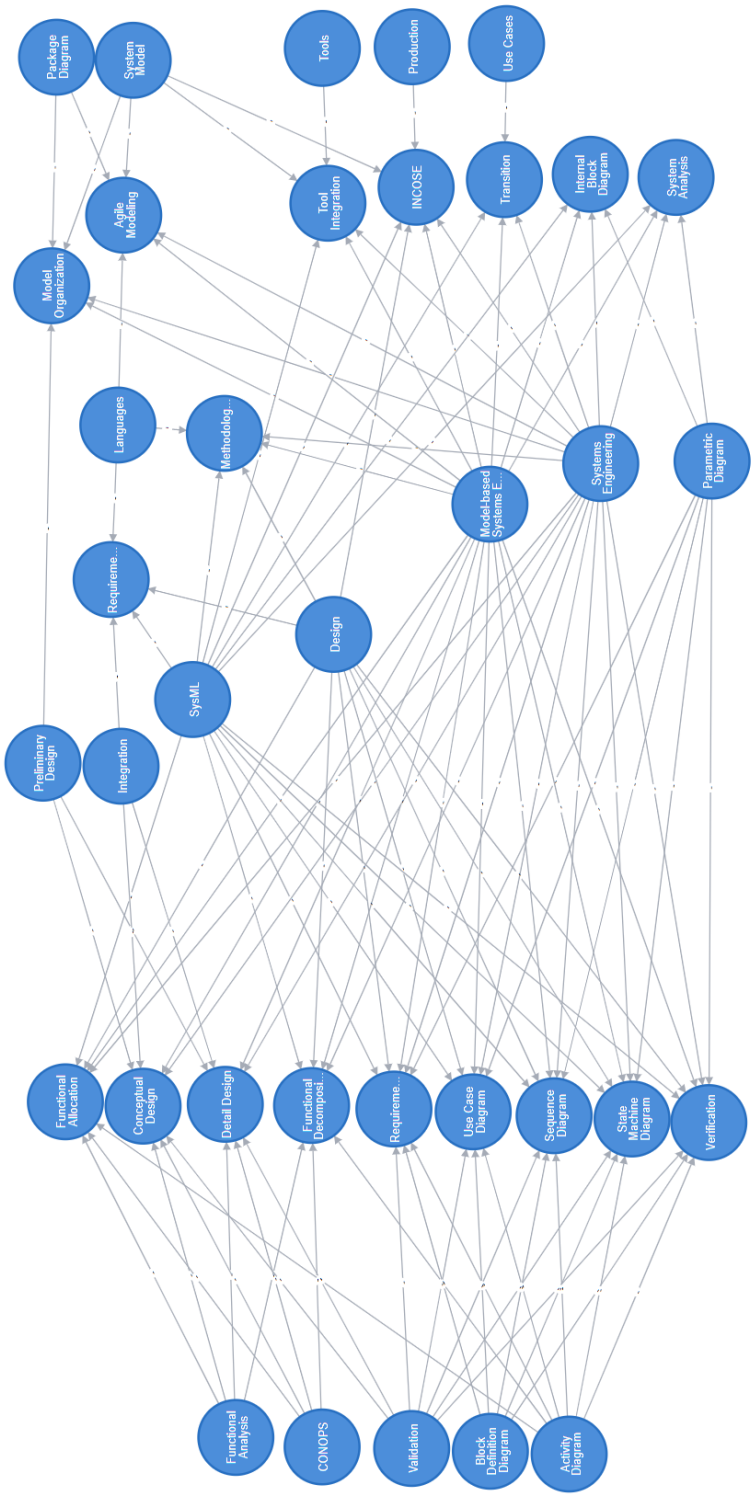

Figure 33: PROF Whitelisting TM for the MBSE Course

4.3.4.2 LDA

The final TM created by the PROF LDA algorithm is for the MBSE course, shown in Figure 34.

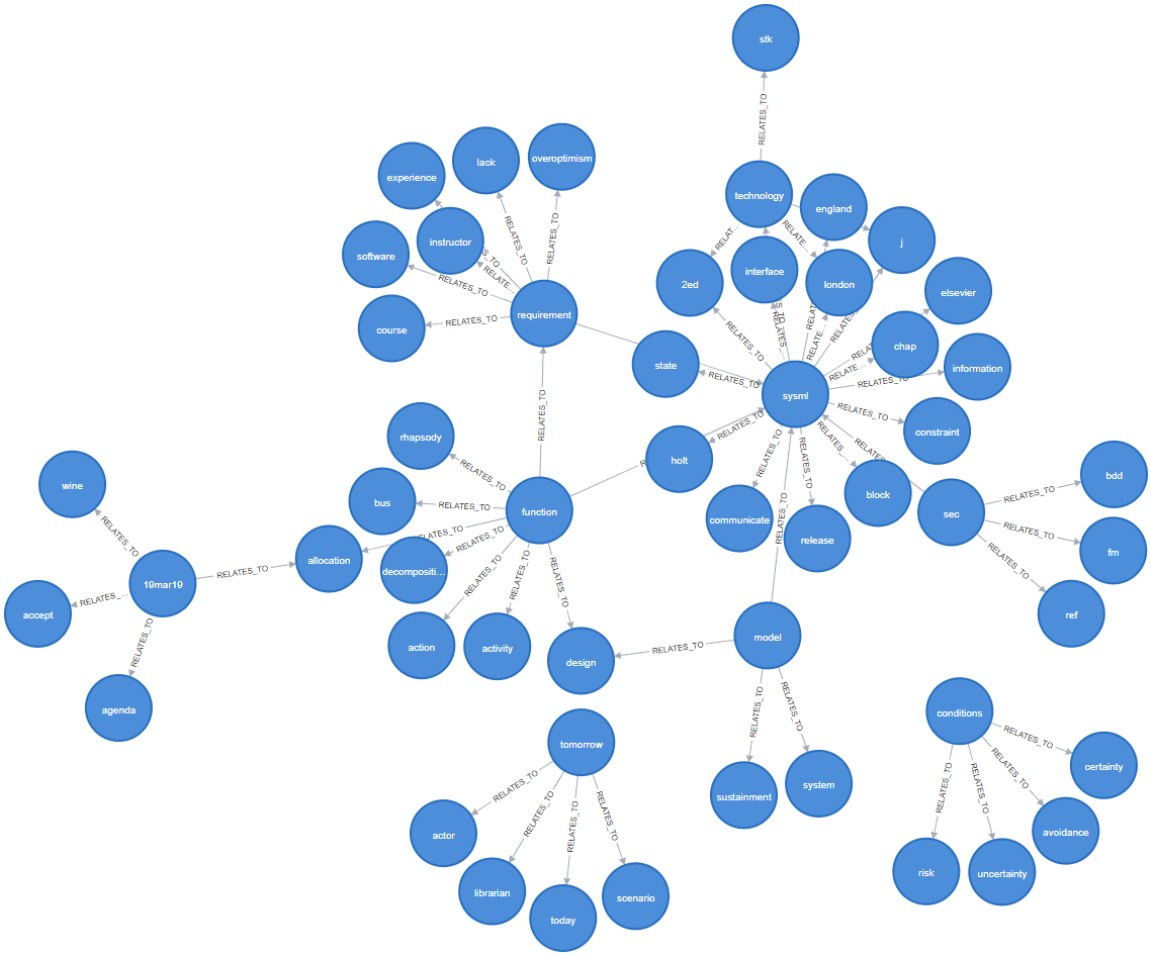

Figure 34: PROF LDA-algorithm TM for the MBSE

4.3.4.3 LDA with ConceptNet

Finally, Figure 35 shows the TM created by this algorithm for the MBSE Course.

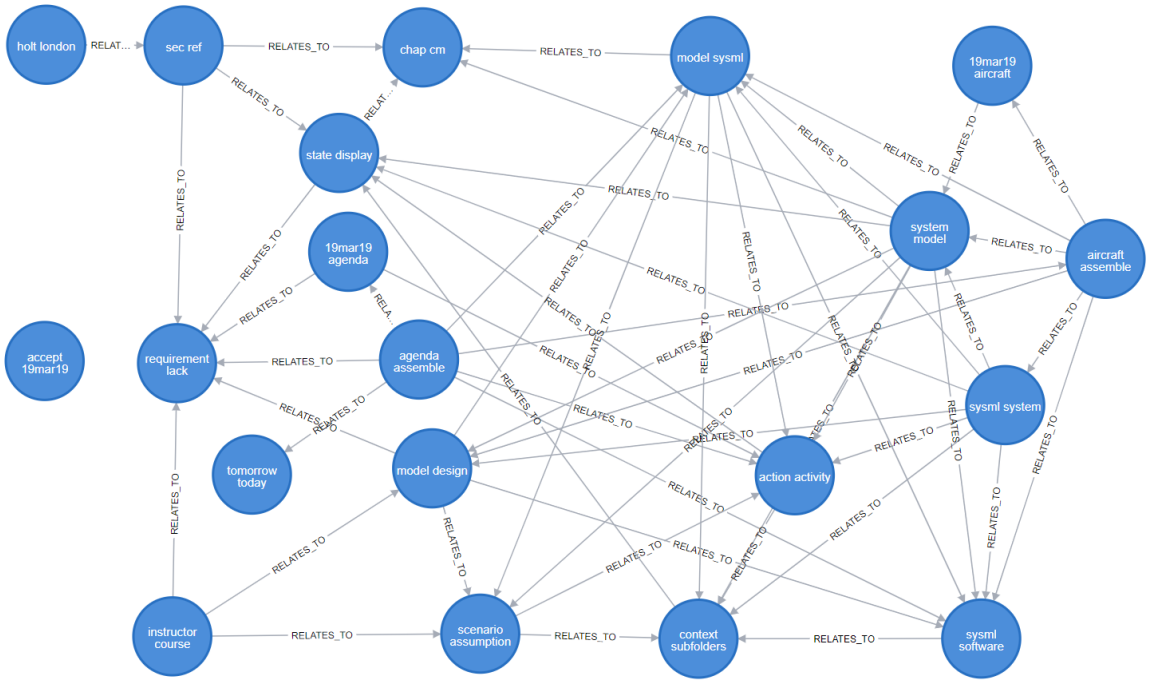

Figure 35: PROF LDA-algorithm with ConceptNet naming TM for the MBSE Course

The third education course and final case study, the MBSE produced the lowest results of the four cases for all three algorithms. There are several reasons for these results, the first of which is the course corpus. This corpus contained a similar amount of material to the other case studies, but consisted of PowerPoint slides and exercises, these do not provide a high likelihood for topic identification compared to the other educational course's textbooks and papers and the NWBC's study guides. These slides and exercises require teacher lecture and guidance, and most likely do not contain all the information for each lesson within its text. Additionally, the SME-Created TM for this course was large and very detailed. The size of the TM gave an opportunity for a higher topic identification rate, but created a low ceiling for the course's McClure score due to the high number of connections and nodes.

4.3.4.4 KSAT

Figure 36 is the KSAT created for the MBSE.

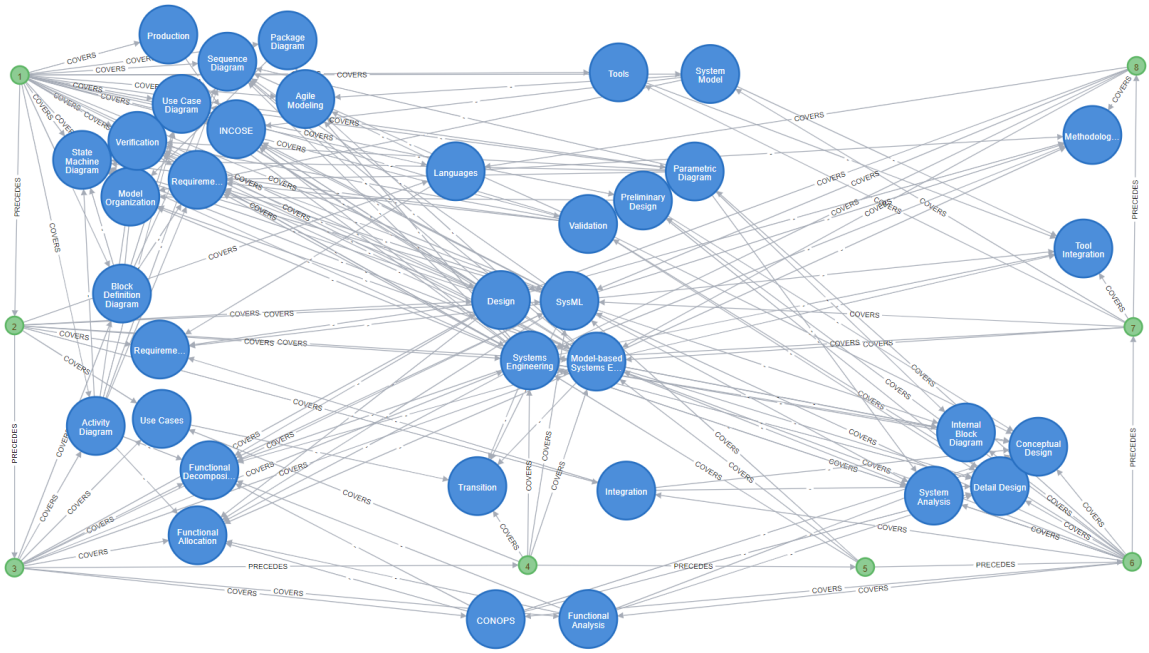

Figure 36: PROF Version 2 KSAT for MBSE

While Figure 36 does contain different groupings of nodes, there are not enough distinct nodes identified to constitute separate learning paths. Many of the nodes in the KSAT are connected to three or more lessons or multiple non-consecutive nodes. In a short training course such as this which only has eight lesson nodes, a node connected to three or more could justify a distinct learning path, but there is no clear separation between these groupings. The individual topics in the clusters in the upper and lower left and lower right are referenced by different lessons than others in their own clusters. These clusters are difficult to separate into a distinct learning path.

4.4 Topic Map Analysis

The first form of analysis that we will explore is the topic identification rate of the different NLP techniques used in the PROF algorithm compared to those included in the SME-Created TMs and course lesson objectives. Table 2 shows the topic identification percentage of each PROF algorithm for each course in the experiment.

Table 2: Topic Identification rate for PROF algorithms against SME-TMs

	Whitelisting	LDA	LDA with ConceptNet
CSCE 660	46%	5.4%	4.1%
SENG 593	81%	41.6%	27.8%
NWBC	51%	15.8%	7.5%
MBSE	31%	8.9%	5.3%

As shown, the rate of identification is highest for the whitelisting algorithm. The algorithm with the next highest rate of topic identification for a majority of the courses is the LDA algorithm, followed by the LDA algorithm with ConceptNet naming as the algorithm with the lowest topic identification rate of the three.

The whitelisting algorithm has the highest topic identification rate. The whitelist contains every topic included in the SME-Created TMs. The reason why the whitelisting algorithm is unable to capture 100% of the topics is due to the text extraction and NLP method used to find the most common nouns and noun chunks. The NLP process does not include all noun chunks and therefore possibly does not include all whitelisted topics present in the text within the listing of topic candidates. Similarly, the SME-identified topics may not exist in the same naming convention within the text, or may only be explicitly referenced a few times.

The LDA algorithm produces topics made up of words that are most likely to be produced by that topic. In generating 20 topics for the corpus of documents, there is an possibility for the algorithm to produce 100 distinct topics. This gives a smaller number of topics identified by the LDA algorithm than by the whitelisting algorithm, but all algorithms identified by the LDA process are added to the TM. To qualify, these topics need to match the SME-Created topics, but cannot be extended to multiple topics on the SME-Created TM. LDA may identify more general topics that are more frequent in the text but may not identify the topics identified in the SME-Created TM that are more specific and mentioned less in the text.

Finally, the LDA and ConceptNet combined algorithm produced the lowest rate of topic identification. The ConceptNet naming process laid out in Chapter III relies on the data available within the ConceptNet system and chooses the two words within each topic most closely related. While this creates a topic name which is possibly more specific than that of the LDA algorithm on its own, it does not create a name that matches to human-created topics like those in the SME-Created TMs. Due to these discrepancies, the LDA and ConceptNet combined algorithm tested produced the lowest rates of topic identification for any of the algorithms tested in the experiment.

The topic identification rate data (Table 2) begins to answer the second investigative question of "How much can we improve topic generation from introducing modern ontological techniques?". In this case, our hypothesis was that LDA would be able to identify similar topics to those identified in the SME TM, However, identification rates did not improve.

The next metric used to evaluate the TMs created by the different stages of the PROF methodology is the McClure graph comparison equation covered in Chapter II. This metric was used to evaluate the different TMs created by each version of PROF for each course with the SME-Created TM used as a baseline. Table 3 shows the

results of this method.

Table 3: Topic Map Comparison using McClure Method

	Whitelisting	LDA	LDA with ConceptNet
CSCE 660	0.12	0	0
SENG 593	0.137	0.140	0.071
NWBC	0.085	0.081	0.020
MBSE	0.057	0	0

The McClure scoring method gives an opportunity to compare like nodes between the SME-Created and PROF-Created TMs. Similar to the topic identification rates between the three algorithms, the highest scoring algorithm was whitelisting, followed by LDA, then LDA with ConceptNet naming. However, there is a higher level of difference between how each course's scores differ per algorithm compared to a consistent decline between all courses in the topic identification rates.

The data in Table 3 answers our first investigative question posed in Chapter I: How closely can computational semantic processing generate a topic map compared with a SME developed one? According to the McClure method, this process can produce a TM with an accuracy between zero and fourteen percent.

The data in Table 2 and Table 3 show the tradeoff between the user input present in the whitelisting algorithm compared to the other two algorithms. While not statistically significant, the data returned by these four studies provides a case for further testing. It seems as though as the time spent by a human to aid in the creation of a TM decreases, the quality of the TM when compared to the SME-created TM also decreases.

The whitelisting algorithm performed greatest among the three different algo-

rithms in both topic identification rate and McClure score. This may mean that the trade of requiring some user time and intervention to create the whitelist may be worth the additional accuracy when creating TMs for a course.

4.5 KSAT Analysis

The KSATs created in the course of this research are transformed version of the TMs created by the whitelisting algorithm. The nodes of the KSATs, similar to the TMs, needed to be manually moved to their position to allow for better readability and clarity to the reader.

For each of the four case studies, a KSAT was created and positioned. It was then visualized and different clusters of nodes, if present, became visible. Table 4 shows the number of learning paths present in each case study's original and, if applicable, altered KSAT.

Table 4: Learning Paths Identified by Visual Inspection

	Initial Course	Modularized Course
CSCE 660	1	3
NWBC	1	5
SENG 593	1	1
MBSE	1	1

While only two of the case studies were able to be altered to create an update KSAT, the results are still promising. It is expected that not every course displays distinct learning paths. The goal of the KSAT is to evaluate courses to update, and to visualize alternate learning paths present in a course, if applicable.

Regarding the research question: "To what extent can we automate the creation

of a KSAT for a course?", the results of this research automates the process except for gathering content, creating a whitelist, and creating a syllabus. While the amount of time and effort to gather content can vary widely, the time required to create a whitelist and syllabus is less than half an hour based on the researcher's experience with each of the case studies.

4.6 Conclusion

The experiment results demonstrate that PROF generates TMs and KSATs with minimal user intervention. While the results are not statistically significant, they provide a baseline where future research can improve topic generation, connection process, and visualization of topics.

V. Conclusions

5.1 Conclusions of Research

The experiment provided data through these four case studies that answers the investigative questions laid out in Chapter I.

The data in Table 2 and Table 3 give insight to the first research question as to how close computational semantic analysis generates a TM compared to a SME-created standard. The whitelisting algorithm provided the most accurate TMs, followed by the LDA algorithm, followed by the LDA algorithm with ConceptNet naming. While the accuracy of this process is not as high as expected, the process provides a technical proof of concept for future research and algorithms to be implemented and tested.

The results also provide insight to the second investigative question. The hypothesis suggested the addition of an ontological technique such as ConceptNet would improve the topic identification rate, but this was not the case as topic identification rate decreased when ConceptNet was introduced.

Finally, the third investigative question was answered and can be further explored in future research. In terms of automating the process of creating a KSAT, this research shows a process for creating a KSAT for a course given only a whitelist of topics and a schedule. The PROF algorithm automates a large portion of this process.

5.2 Significance of Research

This research provides a technical proof of concept for automating the creation of a Topic Map (TM) and Knowledge, Skills, and Abilities Tree (KSAT) for an education or training course. The PROF tool is a method for assessing courses in an attempt to find shorter learning paths present in current Air Force training and education.

These courses can be adapted to self-guided learning and asynchronous environments and altered to more efficiently and effectively educate Airmen.

5.3 Future Work

Future work with this methodology and the algorithms within can take several paths, as well as the application of this process onto other research areas. These paths are as follows:

- Testing additional Natural Language Processing (NLP) methods for topic recognition in the PROF tool and improving on the ConceptNet naming algorithm.
- Optimizing code in the PROF algorithms to perform more efficiently. The code in its current state is viable for a small corpus, but a larger data set may take much longer to process and may use an unnecessary amount of memory.
- Implementing alternative ways to create relationships between topic nodes in the TMs and KSATs and testing for improvement against the data produced in this research.
- Testing additional training and education courses, to include undergraduate and high-school courses to compare topic identification rates in those courses compared to graduate-level courses.
- Building and improving the visualization portion of the PROF algorithm. Creating an interface of some type that automatically separates nodes and adding a method for optionally showing only a smaller subset of relationships between nodes.
- Applying the PROF algorithm to larger and unstructured data sets to attempt to draw out topics and create a TM of broader topics from a less coupled corpus

of data.

- In future work, modified KSATs can be evaluated by SMEs and used to create and edit courses and corpora tested by the PROF tool.

5.4 Summary

This research attempted to define a methodology to better educate, train, and equip the cyber Airman to engage the adversary in multi-domain missions by having access to cutting edge content. The experiment created a methodology and technical proof of concept for automatically creating TMs and semi-automatically creating KSATs and testing this methodology and multiple algorithms against case studies and analyzing the output. The first outputs were three TMs for each of the four courses, with each TM created using a different algorithm. These TMs were compared against SME-Created TMs to test each algorithm's ability to create accurate TMs. While the algorithm that required the most user input was the most successful, the research shows that the process can be fully automated for the other two algorithms. The second experimental output created the KSATs for each course and, if applicable, an updated KSAT for the course visualizing distinct learning paths within the course. These updated KSATs can be used to edit, update, and create new courses to better educate, train, and equip the cyber Airman. While the results shown in this research do not have the statistical significance to draw conclusions, the technical proof of concept and anecdotal evidence shows how semantic analysis and natural language processing can be used to fully automate the process of creating TMs and partially automate the process of creating KSATs.

Appendix A. TIKA Command

```
for /F %i in ('dir /b *.*') do java -jar tika-app-1.21.jar -t %i
>../TIKA_Guides/%i.txt
```

Figure 37: Command to extract text from files in a directory

Appendix B. PROF Code

```
import glob
import spacy
import os
import easygui
import string
from nltk.corpus import stopwords
from nltk.corpus import wordnet as wn
from collections import Counter
from neo4j import GraphDatabase

from nltk.tokenize import RegexpTokenizer
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from gensim import corpora, models
import gensim
import json
from urllib.request import urlopen


def main():
    # Database Credentials
    uri = "bolt://localhost:7687"
    username = "neo4j"
    password = "password"

    # Connect to the neo4j database server
    graphDB_Driver = GraphDatabase.driver(uri, auth=(username,
   password))
    relevant_docs = easygui.fileopenbox(msg="Choose files to
   upload", title="File Upload", filetypes="*.txt",
                                        multiple=True, default='
   :/Users.jorner/Desktop/NWBC - Study Guides/')

    tokenizer = RegexpTokenizer(r'\w+')
    nlp = spacy.load('en_core_web_sm')

    choice = easygui.choicebox(msg="Which algorithm should be run
   ?", title="Alg Choice", choices=["Whitelist", "LDA", "LDACN"])

    sw = stopwords.words("english")
    p_stemmer = PorterStemmer()

    if choice != "LDA" and choice != "LDACN":
        whitelist = easygui.fileopenbox("Whitelist", "Whitelist
   Topics File", filetypes="*.txt",
                                        default='C:/Users/jorner/
   PycharmProject/Thesis/NWBC/Study Guides/Control/topics.txt')

        for f in relevant_docs:
            with open(f, 'r') as file:
                print("Start:", f)
                filename = os.path.basename(file.name).split('.')
   [0]
                doc = nlp(file.read())
```

```
                if choice == "Whitelist":
                    find_topics_whitelist(graphDB_Driver, doc,
filename, whitelist)

        topic_on_topic_connections(graphDB_Driver)
    else:
        texts = []

        for f in relevant_docs:
            with open(f, 'r') as file:
                # print("Start:", f)
                filename = os.path.basename(file.name).split('.')
[0]
                doc = nlp(file.read())

                noun_chunks_full = []
                topic_candidates = []

                for piece in doc.noun_chunks:
                    noun_chunks_full.append(piece.text.lower().
translate(str.maketrans('', '', string.punctuation)))

                for noun_chunk in noun_chunks_full:
                    if "the " in noun_chunk:
                        candidate = noun_chunk[0][4:]
                    elif "a " in noun_chunk:
                        candidate = noun_chunk[0][2:]
                    elif "an " in noun_chunk:
                        candidate = noun_chunk[0][3:]
                    elif "to " in noun_chunk:
                        candidate = noun_chunk[0][3:]
                    elif "this " in noun_chunk:
                        candidate = noun_chunk[0][4:]
                    elif "each " in noun_chunk:
                        candidate = noun_chunk[0][4:]
                    elif "our " in noun_chunk:
                        candidate = noun_chunk[0][3:]
                    elif "their " in noun_chunk:
                        candidate = noun_chunk[0][5:]
                    elif "unclassified " in noun_chunk:
                        candidate = ""
                    elif "\n" in noun_chunk:
                        candidate = ""
                    elif "  s" in noun_chunk:
                        candidate = ""
                    else:
                        candidate = noun_chunk
                    if len(candidate) > 0:
                        topic_candidates.append(candidate)

                topic_candidates = [i for i in topic_candidates
if i not in sw]

                topic_candidates = [i for i in topic_candidates
if not i.isdigit()]
```

```
                topic_candidates = [i for i in topic_candidates
if i.isalnum()]

                with open('D:/Users/jorner/PycharmProjects/Thesis
/LDAStopwords.txt') as stopword_file:
                    customStopwords = stopword_file.read().
split('\n')

                topic_candidates = [i for i in topic_candidates
if i not in customStopwords]

                topic_candidates = [get_lemma(topic_candidate)
for topic_candidate in topic_candidates]

                texts.append(topic_candidates)

        # turn our tokenized documents into a id <-> term
dictionary
        dictionary = corpora.Dictionary(texts)

        # convert tokenized documents into a document-term matrix
        corpus = [dictionary.doc2bow(text) for text in texts]

        # generate LDA model
        ldamodel = gensim.models.ldamodel.LdaModel(corpus=corpus,
 id2word=dictionary, chunksize=2000, alpha='auto',
                                                   eta='auto',
iterations=400, num_topics=20, passes=20, eval_every=None)

        lda_top_topics = ldamodel.top_topics(corpus, topn=5)
        temp = []
        for topic_set in lda_top_topics:
            temp.append(topic_set[0])

        topics = []
        for word_set in temp:
            topic = []
            for word in word_set:
                topic.append(word[1])
            topics.append(topic)

        if choice == "LDA":
            lda_graph_database_load(graphDB_Driver, topics)
        else:
            topic_set = topics
            # Create ConceptNet object and assign names to LDA
topics based on relatedness of LDA topic words within
ConceptNet
            conceptNetTest = conceptNet()
            score_matrix = []
            topic_name = ""
            topics = []
            separated_topics = []
            for topic in topic_set:
                score_bracket = []
                separated_name = ""
```

```
                total_score = 0
                max_score = -1
                # Step through 5 word LDA topic arrays
    combination style 5_C_2
                for i in range(0, len(topic) - 1):
                    for j in range(i + 1, len(topic)):
                        # Query for relatedness score of words in
     ConceptNet
                        relatedness_score = conceptNetTest.
    termsRelatedness(topic[i], topic[j], True)
                        # If max score, save topic strings
                        if relatedness_score > max_score:
                            if topic[i] + " " + topic[j] not in
    topics and topic[j] + " " + topic[i] not in topics:
                                if topic[i] in topic[j]:
                                    topic_name = topic[j] + ""
                                    separated_name = topic[j], ""
                                elif topic[j] in topic[i]:
                                    topic_name = topic[i] + ""
                                    separated_name = topic[i], ""
                                else:
                                    topic_name = topic[i] + " " +
     topic[j]
                                    separated_name = topic[i],
    topic[j]
                                max_score = relatedness_score

                        score_bracket.append(relatedness_score)
                        total_score += relatedness_score

                topics.append(topic_name)
                separated_topics.append(separated_name)
                score_matrix.append(score_bracket)

            temp = []
            for topic in separated_topics:
                if len(topic) != 2:
                    temp.append(topic)
                else:
                    for word in topic:
                        temp.append(word)

            # Create a matrix of each topic name and its
    relatedness to the other topic names
            second_set = 2
            topic_name_similarity_matrix = []
            i = 2
            for pair in separated_topics:
                for half in pair:
                    for word in temp[i:]:
                        topic_name_similarity_matrix.append((half
    , word, conceptNetTest.termsRelatedness(half, word, False)))
                i += 2

            # Average relatedness scores of AC, AD, BC, BD
            averageScores = []
```

```
            averageScoresFull = []
            first = 0
            second = 38
            breaker = 38
            total = 38

            while second < len(topic_name_similarity_matrix):
                scoreA = topic_name_similarity_matrix[first][2]
                scoreB = topic_name_similarity_matrix[second][2]
                scoreC = topic_name_similarity_matrix[first +
1][2]
                scoreD = topic_name_similarity_matrix[second +
1][2]

                averageScore = (scoreA + scoreB + scoreC + scoreD
) / 4
                averageScores.append(averageScore)
                averageScoresFull.append(
                    (averageScore, topic_name_similarity_matrix[
first][0], topic_name_similarity_matrix[second][0],
                     topic_name_similarity_matrix[first][1],
topic_name_similarity_matrix[first + 1][1]))

                first += 2
                second += 2
                if first == total:
                    first += breaker
                    total += breaker
                    breaker -= 2
                    second += breaker
                    total += breaker

            total = 0
            for score in averageScores:
                total += score
            avg = total / len(averageScores)

            # Create a list of ((topic1, topic2), (topic2, topic3
)) where a pair of topics have a relationship with each other
            final_topics = []

            for i in range(0, len(averageScoresFull)):
                if averageScores[i] > avg:
                    topic1 = averageScoresFull[i][1] + " " +
averageScoresFull[i][2]
                    topic2 = averageScoresFull[i][3] + " " +
averageScoresFull[i][4]

                    if topic1.endswith(' '):
                        final_topics.append((topic1[:-1], topic2)
)
                    elif topic1[0].startswith(' '):
                        final_topics.append((topic1[1:], topic2))
                    elif topic2[-1].endswith(' '):
                        final_topics.append((topic1, topic2[:-1])
)
```

```
                    elif topic2[0].startswith(' '):
                        final_topics.append((topic1, topic2[1:]))
                    else:
                        final_topics.append((topic1, topic2))

            lda_conceptnet_graph_database_load(graphDB_Driver,
    topics)
            lda_conceptnet_graph_database_connect(graphDB_Driver,
     final_topics)


def lda_graph_database_load(graphDB_Driver, topics):
    cqlCreate = "CREATE (c:content { name: \"" + "" + "\"})"""
    cqlCreate = cql_create_execute(graphDB_Driver, cqlCreate)

    root_index = 1
    topic = ""
    number = ""
    exists = False

    for topic in topics:
        for word in topic:
            exists = check_if_new_node(word, graphDB_Driver)
            if exists:
                merge_existing_nodes_lda(graphDB_Driver, topic
    [0], word)
            else:
                cqlCreate = "CREATE (t:topic { name: \"" + word +
     "\", number: 0, reference_number: 0, " \

          "relate_num: 0})"
                # print(cqlCreate)
                cql_create_execute(graphDB_Driver, cqlCreate)
                merge_existing_nodes_lda(graphDB_Driver, topic
    [0], word)


def lda_conceptnet_graph_database_load(graphDB_Driver, topics):
    cqlCreate = "CREATE (c:content { name: \"" + "" + "\"})"""
    cqlCreate = cql_create_execute(graphDB_Driver, cqlCreate)

    for topic in topics:
        exists = check_if_new_node(topic, graphDB_Driver)
        if exists:
            merge_existing_nodes_lda(graphDB_Driver, "", topic)
        else:
            cqlCreate = "CREATE (t:topic { name: \"" + topic + "
    \", number: 0, reference_number: 0, " \

          "relate_num: 0})"
            cql_create_execute(graphDB_Driver, cqlCreate)


def lda_conceptnet_graph_database_connect(graphDB_Driver,
    topic_pairs):
     for topic_pair in topic_pairs:
```

```
        firstexists = check_if_new_node(topic_pair[0],
    graphDB_Driver)
        secondexists = check_if_new_node(topic_pair[1],
    graphDB_Driver)
        if firstexists and secondexists:
            merge_existing_nodes_lda(graphDB_Driver, topic_pair
    [0], topic_pair[1])

        elif firstexists and not secondexists:
            print("connect missing1",  topic_pair[0], topic_pair
    [1])
            cqlCreate = "CREATE (t:topic { name: \"" + topic_pair
    [1] + "\", number: 0, reference_number: 0, " \

         "relate_num: 0})"
            cql_create_execute(graphDB_Driver, cqlCreate)
            merge_existing_nodes_lda(graphDB_Driver, topic_pair
    [0], topic_pair[1])
        elif not firstexists and secondexists:
            print("connect missing2", topic_pair[0], topic_pair
    [1])
            cqlCreate = "CREATE (t:topic { name: \"" + topic_pair
    [0] + "\", number: 0, reference_number: 0, " \

         "relate_num: 0})"
            cql_create_execute(graphDB_Driver, cqlCreate)
            merge_existing_nodes_lda(graphDB_Driver, topic_pair
    [0], topic_pair[1])
        else:
            print("connect missing3",  topic_pair[0], topic_pair
    [1])
            cqlCreate = "CREATE (t:topic { name: \"" + topic_pair
    [0] + "\", number: 0, reference_number: 0, " \

         "relate_num: 0})"
            cql_create_execute(graphDB_Driver, cqlCreate)
            cqlCreate = "CREATE (t:topic { name: \"" + topic_pair
    [1] + "\", number: 0, reference_number: 0, " \

         "relate_num: 0})"
            cql_create_execute(graphDB_Driver, cqlCreate)


def get_lemma(word):
    # From https://towardsdatascience.com/topic-modelling-in-
    python-with-nltk-and-gensim-4ef03213cd21
    lemma = wn.morphy(word)
    if lemma is None:
        return word
    else:
        return lemma


def find_topics_whitelist(graphDB_Driver, doc, filename,
    whitelist):
```

```
    # relevant_docs = glob.glob('C:/Users/jorner/Desktop/NWBC -
    Study Guides/TIKA_Guides/*.pdf.txt')
    noun_chunks_root = []
    noun_chunks_full = []
    topic_candidates = []

    for piece in doc.noun_chunks:
        noun_chunks_full.append(piece.text.lower().strip('('))

    word_freq = Counter(noun_chunks_full).most_common(100)

    for word_set in word_freq:
        candidate = word_set[0]
        if "the " in candidate:
            candidate = word_set[0][4:]
        elif "a " in candidate:
            candidate = word_set[0][2:]
        elif "this " in candidate:
            candidate = word_set[0][4:]
        elif "each " in candidate:
            candidate = word_set[0][4:]
        elif "our " in candidate:
            candidate = word_set[0][3:]
        elif "their " in candidate:
            candidate = word_set[0][5:]

        topic_candidates.append((candidate, word_set[1]))



    print("Init sub stop:\t", topic_candidates)

    whitelisted_topics = []
    matched_topics = []

    # Open whitelist file, parse topic names, and save into array
    with open(whitelist, 'r') as whitelisted_topics_file:
        whitelisted_topics_full = whitelisted_topics_file.read().
    split('\n')
        for topic in whitelisted_topics_full:
            whitelisted_topics.append(topic.split('/'))

    #THIS IS FOR PULLING A CSV OF TOPICS
    csv_text = ""
    #END CSV CODE

    # Break topic candidate list into (topic, count)
    for topic_candidate_set in topic_candidates:
        # Set variable to string of topic candidate
        topic_candidate = topic_candidate_set[0]
        # Loop through topic and subname sets
        for whitelisted_topic_set in whitelisted_topics:
            # Loop through subnames of a topic
            for whitelisted_topic in whitelisted_topic_set:
                # Check for candidate and whitelist match
                if topic_candidate == whitelisted_topic:
```

```
                    already_matched = False
                    # Loop through sets of matched topics
                    for matched_topic_set in matched_topics:
                        # If topic has already been added to list
, increment appropriate frequency count
                        if matched_topic_set[0] ==
whitelisted_topic_set[0]:
                            matched_topic_set[1] +=
topic_candidate_set[1]
                            #CSV CODE
                            print("Adding ",whitelisted_topic_set
[0], whitelisted_topic)
                            csv_text += filename + "," +
whitelisted_topic_set[0] + "," + whitelisted_topic + "," + str
(topic_candidate_set[1]) + "\n"
                            #END CSV CODE
                            already_matched = True
                    # If topic has not been added, add topic with
 corresponding frequency count
                    if not already_matched:
                        matched_topics.append([
whitelisted_topic_set[0], topic_candidate_set[1]])
                        #CSV CODE
                        print("New: ", whitelisted_topic_set[0],
whitelisted_topic)
                        csv_text += filename + "," +
whitelisted_topic_set[0] + "," + whitelisted_topic + "," + str
(topic_candidate_set[1]) + "\n"
                        #END CSV CODE

    # CSV CODE
    with open('C:/Users/jorner/Desktop/PROF_CSV.txt', 'a+') as f:
        f.write(csv_text)
    # END CSV CODE

    print("Matched:\t\t", matched_topics)
    print("Root List for processing: ", matched_topics)

    cqlCreate = "CREATE (c:content { name: \"" + filename + "\"})
"""
    cqlCreate = cql_create_execute(graphDB_Driver, cqlCreate)

    root_index = 1
    topic = ""
    number = ""
    exists = False

    for root_set in matched_topics:
        for root in root_set:
            if root_index % 2:
                topic = str(root)
                number = str(root_set[1])
                exists = check_if_new_node(topic, graphDB_Driver)
                if exists:
                    merge_existing_nodes(graphDB_Driver, filename
, topic, number)
```

```
                else:
                    cqlCreate = "CREATE (t:topic { name: \"" +
    topic + "\", number: 0, reference_number: 0, " \
          "relate_num: 0})"
                    # print(cqlCreate)
                    cql_create_execute(graphDB_Driver, cqlCreate)
                    merge_existing_nodes(graphDB_Driver, filename
    , topic, number)
            root_index += 1


def check_if_new_node(node_name, graphDB_Driver):
    """ If topic is new, return 0. If topic is not new, return 1
    """
    cqlMatch = """MATCH (x:topic) WHERE x.name = \"""" +
    node_name + "\" RETURN x"
    return cql_match_execute(graphDB_Driver, cqlMatch)


def cql_create_execute(graphDB_Driver, cqlCommand):
    # Execute the CQL query
    with graphDB_Driver.session() as graphDB_Session:
        # Create nodes
        graphDB_Session.run(cqlCommand)
    return ""


def merge_existing_nodes(graphDB_Driver, filename, topic, number)
    :
    cqlCommand = "MATCH (c:content {name:\"" + filename + "\"}) "
    cqlCommand += "MATCH (t:topic {name:\"" + topic + "\"}) "
    cqlCommand += "MERGE (c)-[r:references]->(t) "
    cqlCommand += "ON CREATE SET r.number = " + number + ", t.
    number = t.number +" + number + ", t.reference_number = "\
                                    "t.reference_number + 1 "
    cqlCommand += "ON MATCH SET r.number = " + number + ", t.
    number = t.number +" + number + ", t.reference_number = " \
                                   "t.reference_number + 1 "
    cql_match_execute(graphDB_Driver, cqlCommand)


def cql_match_execute(graphDB_Driver, cqlCommand):
    """Execute the CQL match command. If nodes are matched,
    return 1, else return 0"""
    # Execute the CQL query
    with graphDB_Driver.session() as graphDB_Session:
        # Create nodes
        nodes = graphDB_Session.run(cqlCommand)
    for node in nodes:
        if "<Record" in str(node):
            return True
        else:
            return False
```

```
    return False


def topic_on_topic_connections(graphDB_Driver):
    """Connect topics to topics to create the TM"""
    cqlCommand = "MATCH(a:topic)<-[:references]-(y:content)\
   nMATCH(b:topic)<-[:references]-(y:content)\n" \
                 "WHERE a.name <> b.name AND a.reference_number >
    1 AND b.reference_number = 1\n" \
                 "MERGE(a)-[:RELATES_TO {sub_name:\"-\"}]-(b)\nON
    CREATE SET a.relate_num = a.relate_num + 1\n" \
                 "ON MATCH SET a.relate_num = a.relate_num + 1\
   nRETURN a, b"
    cql_create_execute(graphDB_Driver, cqlCommand)


def merge_existing_nodes_lda(graphDB_Driver, top_topic, topic):
    cqlCommand = "MATCH (t:topic {name:\"" + top_topic + "\"}) "
    cqlCommand += "MATCH (s:topic {name:\"" + topic + "\"}) "
    cqlCommand += "WHERE t.name <> s.name "
    cqlCommand += "MERGE (t)-[r:RELATES_TO {sub_name:\"-\"}]-(s)
   "
    cqlCommand += "ON CREATE SET r.number = 0 "
    cqlCommand += "ON MATCH SET r.number = 0"
    cql_match_execute(graphDB_Driver, cqlCommand)

#Class to access and work with MIT's ConceptNet web API
#Original Author: fito_segrera / http://fii.to
#Adapted by Jacob Orner for work with PROF tool
class conceptNet:

    def __init__(self):
        self.url = "http://192.168.195.128/"

    def lookup(self, term, verbose):
        """Lookup a topic in ConceptNet. Print a list of
   information on the topic"""
        url_to_search = self.url + "c/en/" + term
        data = urlopen(url_to_search)
        json_data = json.load(data)
        if verbose:
            print(url_to_search)
            for i in json_data["edges"]:
                print("-----------------")
                print(i["end"])
                print("relation:", i["rel"])
                print(i["end"])
                print(i["start"])
                print("weight:", i["weight"])

    def find_related(self, topic, verbose):
        """Find the related topics top this in ConceptNet. Return
    a list of related topics"""
        url_to_search = self.url + "related/c/en/" + topic + "?
   filter=/c/en"
        data = urlopen(url_to_search)
```

```
        json_data = json.load(data)
        if verbose:
            print(json_data["@id"])
            for i in json_data["related"]:
                print("--------------------")
                print(i["@id"][6:])
                print("weight:", i["weight"])

    def termsRelatedness(self, term1, term2, verbose):
        url_to_search = self.url + "relatedness?node1=/c/en/"+
   term1 + "&node2=/c/en/" + term2
        data = urlopen(url_to_search)
        json_data = json.load(data)

        return json_data["value"]


    def termsAssociation(self, term1, term2, verbose):
        url_to_search = self.url + "query?node=/c/en/" + term1 +
   "&other=/c/en/" + term2
        data = urlopen(url_to_search)
        json_data = json.load(data)
        if verbose:
            for i in json_data["edges"]:
                print("--------------------")
                print(i["@id"])

if __name__ == '__main__':
    main()

# PROF - Pedagogical Resource Organization Framework
```

Listing B.1: Full PROF Code

Appendix C. KSAT Creation Code

```
import glob
import easygui
from py2neo import Graph, Node, Relationship, Database,
    NodeMatcher


def main():
    db = Database("bolt://localhost:7687")
    g = Graph(database=db, password="password")
    tx = g.begin()

    lesson_material = []
    s = easygui.fileopenbox("Schedule", "Select Schedule File",
    filetypes="*.txt",
                                        default='C:/Users/jorner/
    PycharmProject/Thesis/')
    with open(s, 'r') as schedule:
        for lesson in schedule:
            data = lesson.split(',')
            lesson_material.append([d.strip() for d in data])

        number_of_lessons = len(set(x[0] for x in lesson_material
    ))

        relevant_docs = glob.glob('./class_files/*.txt')


        # Upload to neo4j database
        selector = NodeMatcher(g)

        tx = g.begin()
        a = Node("Lesson", name="1")
        tx.create(a)
        tx.commit()

        for lesson, reading in lesson_material[1:]:
            tx = g.begin()
            existing_node = selector.match("Lesson", name=str(
    lesson)).first()
            if existing_node:
                b = existing_node
            else:
                b = Node("Lesson", name=str(lesson))
                tx.create(b)

            if a != b and lesson != "1":
                c = Relationship(a, "PRECEDES", b, length=2)
                tx.create(c)
            tx.commit()
            a = b

        # Draw nodes for lessons to readings
        for lesson, reading in lesson_material:
            if reading is not "":
```

```
                tx = g.begin()
                existing_node = selector.match("Lesson", name=
    lesson).first()
                if existing_node:
                    a = existing_node
                else:
                    a = Node("Lesson", name=lesson)
                    tx.create(a)

                existing_node = selector.match("content", name=
    reading).first()
                if existing_node:
                    b = existing_node
                else:
                    b = Node("content", name=reading)
                    tx.create(b)

                c = Relationship(a, "ASSIGNS", b)
                tx.create(c)
                tx.commit()


if __name__ == '__main__':
    main()
```

Listing C.1: KSAT Creation Code

Sum of Count	Column Labels																
Row Labels	Xia	Varol	Toorani	Tan	Tam	Scoccia	NIST Chapters 3,4,5	NIST Chapter 2	Mutchler	Montenegro	Mondal	Jaiswal	Falliere	Elbez	Assante	Arnatovich	Grand Total
AES										12							12
Android	22			72	68	7			18	14		5				6	212
App Store						3											3
Apps	25	3		62	39	21	27	27	60	13	1	10				30	318
C#													6				6
C++													6				6
Camera		2				4											6
CDMA											2						2
Cell Phones		8	1														9
Cellular		6		8	20				3	2	1						40
Data	4	1		6	20	7	65	65	2	8			18	8	3	6	213
eMail															3		3
Encryption			5		4		19	19		4							51
Espionage															3		3
Firewall							78	78									156
Free Apps						4			2								6
GSM			18								2						20
Internet	2		4		5		15	15	2		3		4	2	2		54
iOS		1															1
Java				4												5	9
LAN			1										3				4
Networks		1	16		15		80	80			1		3	6	6		208
Passwords							22	22									44
PII		1															1
PIN			4														4
REST										3		1					4
Servers				3													3
Social Media	1																1
TCP-IP											3						3
Traffic							17	17			1						35
Updates							10	10					6				26
Viruses		2															2
VoIP											3						3
Worms													3				3
Grand Total	**54**	**25**	**49**	**155**	**171**	**46**	**333**	**333**	**87**	**56**	**17**	**16**	**49**	**16**	**17**	**47**	**1471**

Figure 38: PROF Version 2 Topic Reference Counts Per Document

Appendix D. CQL Code

```
MATCH (l:LESSON)-[:ASSIGNS]→(c:content)
MATCH (c:content)-[:references]→(t:topic)
MERGE (l)-[:COVERS]→(t)
RETURN l,t
```

Figure 39: CQL Statement to Link Lesson and Topic Nodes

Bibliography

1. W Newhouse, S Keith, B Scribner, and G Witte. National Initiative for Cybersecurity Education (NICE) Cybersecurity Workforce Framework, Special Publication 800-181. *NIST*, 2017.

2. Travis Ralph. Air Force Cyber Education and Training. Accessed: 2019-10-14, 2019.

3. Chaitra M Hardison, Leslie Adrienne Payne, John A Hamm, Angela Clague, Jacqueline Torres, David Schulker, and John S Crown. *Attracting, Recruiting, and Retaining Successful Cyberspace Operations Officers Cyber Workforce Interview Findings*. RAND Corporation, 2019.

4. Pilot Training Next. `https://www.aetc.af.mil/About-Us/Pilot-Training-Next/`. Acessed: 2019-08-01.

5. Stephen Losey. The Air Force is Revolutionizing the Way Airmen Learn to be Aviators. `https://www.airforcetimes.com/news/your-air-force/2018/09/30/the-air-force-is-revolutionizing-the-way-airmen-learn-to-be-aviators/`, Oct 2018.

6. USA Jobs. FAQs. `https://www.usajobs.gov/Help/faq/job-announcement/KSAs/`, 2019. Accessed: 2019-09-10.

7. John R. McClure, Brian Sonak, and Hoi K. Suen. Concept Map Assessment of Classroom Learning: Reliability, Validity, and Logistical Practicality. *Journal of Research in Science Teaching*, 36(4):475–492, 1999.

8. Cyber Education Hub. `https://www.afit.edu/ceh`. Accessed: 2020-01-17.

9. Khan Academy: Free Online Courses, Lessons, and Practice. https://www.khanacademy.org/. Accessed: 2020-01-17.

10. Microsoft. Microsoft word. https://products.office.com/en-us/word. Accessed: 2020-01-17.

11. Microsoft. Microsoft powerpoint. https://products.office.com/en-us/powerpoint. Accessed: 2020-01-17.

12. E Taft, J Pravetz, Stephen Zilles, and Larry Masinter. The Application/PDF Media Type. *Internet proposed standard RFC*, 3778, 2004.

13. Landon G M Tomcho. Experimentation and Analysis Using Modern Gamification. Master's thesis, Air Force Institute of Technology, 2019.

14. Valery Vodovozov and Zoja Raud. Concept Maps for Teaching, Learning, and Assessment in Electronics. *Education Research International*, 2015, 2015.

15. Frank Mayadas. Asynchronous Learning Networks: A Sloan Foundation Perspective. *Journal of Asynchronous Learning Networks*, 1(1):1–16, 1997.

16. Bill Tucker. The Flipped Classroom. *Education Next*, 12(1):82–83, 2012.

17. Janet Bailey and Robert B Mitchell. Industry Perceptions of the Competencies Needed by Computer Programmers: Technical, Business, and Soft Skills. *Journal of Computer Information Systems*, 47(2):28–33, 2006.

18. A. Parrish. Long-term View Needed for Cybersecurity Education. *Insid. High. Ed*, 2017:1–10, 2017.

19. Chad Dacus. *Cyber Education and Training*. PhD thesis, Air Force Cyber College, 2013.

20. R. M. Abrams. The U.S. Military and Higher Education: A Brief History. *The ANNALS of the American Academy of Political and Social Science*, 502(1):15–28, 1989.

21. Anju Relan and Bijan B Gillani. Web-based Instruction and the Traditional Classroom: Similarities and Differences. *Web-based instruction*, 62:41–46, 1997.

22. Stefan Hrastinski. Asynchronous and Synchronous E-Learning. *EDUCAUSE Quarterly*, 31(4):51–55, 2008.

23. Benjamin S Bloom. *Taxonomy of educational objectives. Vol. 1: Cognitive domain*, volume 1. Univ. Chicago Press, 1956.

24. Jacqueline O Flaherty and Craig Phillips. Internet and Higher Education: The use of Flipped Classrooms in Higher Education: A Scoping Review. *The Internet and Higher Education*, 25:85–95, 2015.

25. Alan D. Carswell and Viswanath Venkatesh. Learner Outcomes in an Asynchronous Distance Education Environment. *International Journal on Human-Computer Studies*, 56:475–494, 2002.

26. Joddy Murray. Composing Multimodality. *Multimodal Composition: A Critical Sourcebook. Boston: Bedford/St. Martins*, 2013.

27. Charles Fadel and Cheryl Lemke. Multimodal Learning through Media: What the research says. *San Jose, CA: Cisco Systems*, pages 1–24, 2008.

28. S Hazari. Applying Instructional Design Theories to Improve Efficacy of Technology-assisted presentations. *Journal of Instruction Delivery Systems*, 18(2):24–33, 2004.

29. Seth A. Martin. Unguided Cyber Education Techniques of the Non-Expert. Master's thesis, Air Force Institute of Technology, 2019.

30. Athanasios Hatzigaidas, Anastasia Papastergiou, George Tryfon, and Despoina Maritsa. Topic Map Existing Tools: A Brief Review. *International Conference on Theory and Applications of Mathematics and Informatics - ICTAMI 2004*, pages 185–201, 2004.

31. Fakhre Alam, Shaukat Ali, Muhammad Abid Khan, Shah Khusro, and Azhar Rauf. A Comparative Study of RDF and Topic Maps Development Tools and APIs. *Bahria University Journal of Information & Communication Technologies*, 7(1):1–12, 2014.

32. Krunoslav Zubrinic, Damir Kalpic, and Mario Milicevic. The Automatic Creation of Concept Maps from Documents Written Using Morphologically Rich Languages. *Expert Systems with Applications*, 39(16):12709–12718, 2012.

33. Jorge J. Villalon and Rafael A. Calvo. Concept Map Mining: A Definition and a Framework for its Evaluation. In *Proceedings - 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Workshops, WI-IAT Workshops 2008*, pages 357–360, 2008.

34. Nancy A. Letassy, Melissa S. Medina, Mark L. Britton, Vince Dennis, and Jolaine R. Draugalis. A Progressive, Collaborative Process to Improve a Curriculum and Define an Assessment Program. *American Journal of Pharmaceutical Education*, 79(4):1–5, 2015.

35. Yu Shih Lin, Yi Chun Chang, Keng Hou Liew, and Chih Ping Chu. Effects of Concept Map Extraction and a Test-Based Diagnostic Environment on Learning

Achievement and Learners' Perceptions. *British Journal of Educational Technology*, 47(4):649–664, 2016.

36. YoungJu Kang and Albert D. Ritzhaupt. A Job Announcement Analysis of Educational Technology Professional Positions. *Journal of Educational Technology Systems*, 43(3):231–256, 2015.

37. Christine Kay and Eliza Moncarz. Knowledge, Skills, and Abilities for Lodging Management Success. *Cornell Hotel and Restaurant Administration Quarterly*, 45(3):285–298, 2004.

38. Marcel van der Klink and Jo Boon. The Investigation of Competencies Within Professional Domains. *Human Resource Development International*, 5(4):411–424, 2002.

39. A.J. Cañas and J. Novak. Itineraries: Capturing Instructors' Experience Using Concept Maps as Learning Object Organizers. In *Fourth Int. Conference on Concept Mapping*, volume October 2010, 2010.

40. Christopher D. Manning and Hinrich Schutze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

41. Julian Kupiec, Jan Pedersen, and Francine Chen. A Trainable Document Summarizer. *Xerox Palo Alto Research Center*, 1999.

42. Thomas K Landauer and Susan T Dumais. A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge. *Psychological review*, 104(2):211, 1997.

43. Thomas Hofmann. Probabilistic Latent Semantic Analysis. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 289–296. Morgan Kaufmann Publishers Inc., 1999.

44. Nick C Ellis. Frequency Effects in Natural Language Processing. *Studies in Second Language Acquisition*, pages 143–188, 2002.

45. David M Blei, Andrew Y Ng, and Michael I Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

46. Jonathan J Webster and Chunyu Kit. Tokenization as the Initial Phase in NLP. In *COLING 1992 Volume 4: The 15th International Conference on Computational Linguistics*, 1992.

47. Joël Plisson, Nada Lavrac, Dunja Mladenic, et al. A Rule Based Approach to Word Lemmatization. *Proceedings of IS-2004*, pages 83–86, 2004.

48. Thomas K Landauer, Danielle S McNamara, Simon Dennis, and Walter Kintsch. *Handbook of Latent Semantic Analysis*. Psychology Press, 2013.

49. Heeyeul Kwon, Jieun Kim, and Yongtae Park. Applying LSA Text Mining Technique in Envisioning Social Impacts of Emerging Technologies: The Case of Drone Technology. *Technovation*, 60:15–28, 2017.

50. Peter W Foltz, Darrell Laham, and Thomas K Landauer. Automated Essay Scoring: Applications to Educational Technology. In *EdMedia+ Innovate Learning*, pages 939–944. Association for the Advancement of Computing in Education (AACE), 1999.

51. Debra Trusso Haley, Pete Thomas, Anne De Roeck, and Marian Petre. A Research Taxonomy for Latent Semantic Analysis-based Educational Applications. In *International Conference on Recent Advances in Natural Language Processing*, volume 5, pages 575–579, 2005.

52. Leonhard Hennig. Topic-Based Multi-Document Summarization with Probabilistic Latent Semantic Analysis. In *Proceedings of the International Conference RANLP-2009*, pages 144–149, 2009.

53. Thomas Hofmann. Probabilistic Latent Semantic Indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 50–57, 1999.

54. Anna Bosch, Andrew Zisserman, and Xavier Muñoz. Scene Classification via pLSA. In *European conference on computer vision*, pages 517–530. Springer, 2006.

55. Joost Van De Weijer, Cordelia Schmid, Jakob Verbeek, and Diane Larlus. Learning Color Names for Real-world Applications. *IEEE Transactions on Image Processing*, 18(7):1512–1523, 2009.

56. Gui-Rong Xue, Wenyuan Dai, Qiang Yang, and Yong Yu. Topic-bridged PLSA for Cross-domain Text Classification. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 627–634. ACM, 2008.

57. Hamed Jelodar, Yongli Wang, Chi Yuan, Xia Feng, Xiahui Jiang, Yanchao Li, and Liang Zhao. Latent Dirichlet Allocation (LDA) and Topic modeling: Models, Applications, a Survey. *Multimedia Tools and Applications*, 78(11):15169–15211, 2019.

58. SIL. Lexical database definition. `https://glossary.sil.org/term/lexical-database`. Accessed: 2020-01-16.

59. Torsten Zesch, Christof Müller, and Iryna Gurevych. Using Wiktionary for Computing Semantic Relatedness. In *AAAI*, volume 8, pages 861–866, 2008.

60. Christof Müller and Iryna Gurevych. Using Wikipedia and Wiktionary in Domain-specific Information Retrieval. In *Workshop of the Cross-Language Evaluation Forum for European Languages*, pages 219–226. Springer, 2008.

61. Joshua Chin Robyn Speer and Catherine Havasi. ConceptNet 5.5: An Open Multilingual Graph of General Knowledge. *AAAI 31*, 2017.

62. Apache TIKA A Content Analysis Toolkit. `https://tika.apache.org/`. Accessed: 2019-05-01.

63. Jacob Orner and Richard Dill. PROF : An Assistive Tool for Educators to Semi-Autonomously Create Topic Maps and Skill Trees. In *Special Interest Group on Information Technology Education*, page 141, Tacoma, WA, 2019.

64. ANSI. Iso-ir-006: Ascii graphic character set. `http://asciidoc.org/`, December 1, 1975. Accessed: 17-December-2019.

65. Spacy Industrial-Strength Natural Language Processing. `https://spacy.io/`. Accessed: 2019-05-10.

66. Neo4j graph database platform. `https://neo4j.com/`. Accessed: 2019-05-03.

67. R Rehurek and P Sojka. Gensim–Python Framework for Vector Space Modeling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, 3(2), 2011.

68. Python-ConceptNet, December 6, 2015.

Acronyms

AETC Air Education and Training Command. 1, 2

AF Air Force. iv, 1, 2

AFIT Air Force Institute of Technology. 4, 39

AFLSE Air Force Learning Services Ecosystem. 2

API Application Programming Interface. 28

ASCII American Standard Code for Information Interchange. 26, 27

CEH Cyber Education Hub. 4

CoL Continuum of Learning. 2

CQL Cypher Query Language. 31, 32, 35

CSCE Computer Science and Computer Engineering. 39

DoD Department of Defense. iv, 2, 8, 1

KSA Knowledge, Skills, and Abilities. iv, 2, 8, 12, 13, 14, 15, 22

KSAT Knowledge, Skills, and Abilities Tree. iv, 2, 3, 4, 5, 6, 7, 8, 10, 14, 17, 18, 20, 21, 22, 23, 25, 28, 33, 35, 36, 37, 38, 39, 58, 70, 72, 1

LDA Latent-Dirichlet Allocation. iv, 5, 18, 20, 23, 25, 29, 36, 39, 52, 56, 68, 1

LSA Latent Semantic Analysis. 18, 19, 20, 29

MBSE Model-Based Systems Engineering. 4, 39, 63

NICE National Initiative for Cybersecurity Education. iv

NICE Framework National Initiative for Cybersecurity Education Cybersecurity Workforce Framework. 13, 14

NIST National Institute of Standards and Technology. 13

NLP Natural Language Processing. 1, 2, 5, 8, 18, 19, 20, 22, 23, 28, 36, 52, 73

NWBC Network Warfare Bridge Course. 4, 39, 53, 56, 58

PDF Portable Document Format. 5, 24, 27

pLSA Probabilistic Latent Semantic Analysis. 18, 20, 29

PROF Pedagogical Resource Organization Framework. 2, 4, 5, 6, 7, 8, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 35, 36, 37, 39, 56

SENG Systems Engineering. 39

SME Subject-Matter Expert. iv, 5, 11, 14, 23, 24, 25, 36, 37, 38, 39, 40, 41, 68, 69

TM Topic Map. iv, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 15, 16, 17, 18, 20, 21, 22, 23, 24, 25, 27, 28, 30, 32, 33, 35, 36, 37, 38, 39, 40, 41, 42, 43, 49, 52, 56, 63, 68, 69, 72, 1

USAF United States Air Force. iv, 1

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
26–03–2020	Master's Thesis	Sept 2018 — Mar 2020

4. TITLE AND SUBTITLE

Applying Data Organizational Techniques
to Enhance Air Force Learning

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Jacob A. Q. Orner, 2d Lt, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-MS-20-M-052

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Intentionally Left Blank

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The USAF and the DoD use traditional schoolhouses to educate and train personnel. The physical aspects of these schoolhouses limit throughput. A method to increase throughput is to shift towards an asynchronous learning environment where students move through content at individually. This research introduces a methodology for transforming a set of unstructured documents into an organized TM students can use to orient themselves in a domain. The research identifies learning paths within the TM to create a directed KSAT. We apply this methodology in four case studies, each an education or training course. Using a graph comparison metric and the topic identification rates for the TMs, we tested a whitelisting algorithm to identify topics with up to 81% accuracy, and leveraged a standalone LDA algorithm and the same LDA algorithm with ConceptNet for topic naming. The research also produced a KSAT for all case studies and two modified KSATs. The research shows that TMs and KSATs can automatically be created with minimal user input. This methodology could help increase throughput in Air Force education and training pipelines.

15. SUBJECT TERMS

Self-guided learning; Asynchronous Learning; Topic Maps; Knowledge, Skills, and Abilities Trees; Data Organization

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Major Richard Dill, AFIT/ENG
U	U	U	UU	113	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636, ext 3652; richard.dill@afit.edu