Using Topic Modeling to Extract Pre-Service Teachers' Understandings of Computational Thinking from Their Coding Reflections

Maria Cutumisu and Qi Guo

***Abstract*—*Contribution:* This study employs the automatic scoring of short essays as a novel way to determine pre-service teachers' knowledge of and attitudes towards computational thinking (CT) from their written reflections. Implications about designing CT courses for pre-service teachers are discussed. *Background:* CT is an essential 21st-century competency that supports the development of problem-solving skills. Inspired by computing science problem-solving practices, CT should transcend disciplines, but few universities or colleges include CT courses or CT content in their core courses. It is also difficult to know what pre-service teachers think about CT and their role in promoting it. *Research Questions:* Do pre-service teachers' coding reflections reveal any important information about their knowledge of, skills in, and attitudes towards computational thinking? *Methodology:* Traditional qualitative techniques based on human raters are impractical in analyzing hundreds of essays. Topic modeling, an unsupervised machine learning modeling technique, was employed to extract topical features from participants' reflections. In one section of an undergraduate Introduction to Educational Technology course offered at a large university in Western Canada, n = 139 pre-service teachers wrote a short reflection on their experience following a 20-hour Accelerated Intro to Computer Science Code.org course. Topics were identified by analyzing contextual trends in participants' written reflections. *Findings:* Results showed that pre-service teachers' reflections included CT concepts, practices, and perspectives. Specifically, participants connected the coding activity to prior knowledge and experiences.**

***Index Terms*—Computational thinking, computing skills, educational technology, higher education, pre-service teachers, problem solving, programming, topic modeling**

I. Introduction

ADDRESSING complex contemporary societal issues increasingly requires innovative problem-solving skills, interdisciplinarity, and technological expertise [1]. Computational thinking (CT), a 21st-century competency that promotes a powerful new way of thinking inspired by computing science [2], [3], holds the promise of preparing learners to solve these types of complex problems. CT is defined as a set of problem-solving thought processes that can be carried out by an information processing agent, and is considered a fundamental skill that everyone will use by the middle of the 21st century [4]-[8] and that will help thinkers from all disciplines [6], [9], [10].

Integrating CT across the K-12 curriculum is imperative [11], as it has deep ramifications for developing knowledgeable, innovative future generations. It is believed that CT is most effective when introduced at an early age (i.e., in elementary and early secondary education), preferably before students learn a programming language [12]. Also, STEM career choices have been associated with interest in STEM fields in middle school [13] and high school [14]. One way to ensure a smooth integration is to expose teachers to CT principles in teacher education courses [15]. Although several research studies conducted recently have focused on K-12 students, it is still not known to what extent pre-service teachers are prepared to teach and model CT skills and strategies to their students, especially as CT is already becoming increasingly integrated in the K-12 curriculum around the world [16]. Moreover, a review of teaching and learning CT revealed an acute need to prepare teachers to integrate CT into their pedagogy [17], particularly with respect to integrating CT into classroom practices [18], [19]. Additionally, most CT teacher interventions are conducted for in-service computing science teachers, with a few notable exceptions [20]. Positive results regarding the perception and understanding of CT were reported in some intervention studies with in-service teachers [21], [22]. Lastly, few studies have rigorously examined the assessment of CT [23].

More interventions geared towards non-computing science majors are needed, if CT is to be adopted across different subjects in K-12 [17]. However, even after such interventions are conducted, examining the computational content of the written samples of large groups of pre-service teachers manually is time consuming, and can be subjective and error prone. Without specialized tools to extract meaning and themes from such large-scale text collections, it is difficult to understand the general themes that could emerge from CT interactions. Moreover, every new written reflection can be identified with more than one theme, which makes manual tasks

Corresponding author: M. Cutumisu. This work was supported in part by the Social Sciences and Humanities Research Council of Canada - Insight Development Grant (SSHRC IDG) RES0034954, the Natural Sciences and Engineering Research Council (NSERC DG), and the Killam Cornerstone Operating Grant RES0043207.

M. Cutumisu is an assistant professor at the University of Alberta, Edmonton, T6G 2G5 Canada (e-mail: cutumisu@ualberta.ca).

Q. Guo is a postdoctoral fellow at the University of Alberta, Edmonton, T6G 2G5 Canada (e-mail: qig@ualberta.ca).

of theme extraction even more impractical. Assessing the strengths and weaknesses of pre-service teachers' understanding of CT is an important consideration, if CT is to be adopted widely in the K-12 curricula. Assessing open-ended student work automatically in a post-secondary educational environment has become increasingly important as the budget for augmenting instructional teams to mark student assessments has shrunk. This study aimed to understand and characterize the CT understandings of a large sample of pre-service teachers by examining their written reflections after having completed the Accelerated Intro to Computer Science course offered by Code.org. This type of activity is often included in the Hour of Code (hourofcode.com) annual event held in over 180 countries during the Computer Science Education Week, to spark individuals' interest and participation into computing science and to demystify and popularize coding. This research poses the following question: *Do pre-service teachers' coding reflections reveal any important information about their knowledge, skills, and attitudes on computational thinking?*

To answer this question, topic modeling was proposed as an exploratory educational research tool to make sense of pre-service teachers' written reflections. Topic models are used to identify topics or themes that emerge from a collection of documents by analyzing the statistical distribution of words across these documents. The advantage of the topic model approach is its automated ability to reflect the properties of written documents, to extract topics that explain the dataset, and to answer questions based on large-scale written productions in a low-cost, timely, and efficient manner, without human intervention. This approach discovers topics in pre-service teachers' reflections, aiming to enhance current understandings of the common themes they use to express their thoughts about CT. Increased educator awareness of the importance of teaching CT to students can lead to increased opportunities for students to engage meaningfully with their digital environments.

II. Theoretical Framework and Related Work

This research draws on the theoretical framework of topic modeling to guide the data analysis in this study. Topic modeling derives from the idea that texts include a hidden topic structure or a latent variable that causes a departure from a random expected word usage [24]. A topic modeling algorithm is a tool that helps uncover these hidden structures embedded in text. As the main topic in this work is exploring pre-service teachers' views on CT, this section focuses on the conceptual framework underpinning CT, and the related applications of topic modeling in education.

A. Computational Thinking

Computational thinking (CT) is regarded as a unique set of problem-solving processes that should be a part of all K-12 students' analytical ability in the 21st century, with applications well beyond computing. The CT conceptual framework falls under the computing paradigm that highlights cognitive abilities focused on information processing [25], which include problem-solving skills. Specifically, information processing tasks can be solved in a systematic way by employing logical reasoning and abstract thinking [26]. In 2006, Jeannette Wing drew attention to CT [5], later redefining it as an approach to "solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science" [6]. She expanded CT as "the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" [27]. In addition to computation or algorithmic thinking, CT skills include core concepts from computing science: subgoaling (breaking down complex problems into simpler parts or decomposition), abstraction (deciding which problem details are essential), and data representation (displaying data in different forms)—all of which can be carried out with or without a machine [28]. More generally, the four pillars of CT are considered to be decomposition, abstraction, algorithmic thinking, and debugging [6], [15], [20], [28], [29].

CT was also articulated as a three-dimensional competency [3], [30] composed of concepts (e.g., data, sequencing, loops, conditionals, etc.), practices (iterative development, testing and debugging, reusing, and abstracting and modularizing), and perspectives (e.g., interpersonal and intrapersonal relationships). This approach, adopted in this paper, was informed by the CS Principles of CT of the College Board and the National Science Foundation (NSF), developed around the seven big ideas of computing: creativity, abstraction, data, algorithms, programming, interconnectivity, and innovation.

Researchers have long underscored the importance of CT at the post-secondary level [31], [32]. Recently, several studies were conducted to ascertain teachers' level of understanding of CT. Previously, a study that designed and assessed CT modules for pre-service teachers found that introducing CT in education courses can aid participants' understanding of CT concepts [20]. Even in general undergraduate populations, the introduction of CT in science courses was shown to improve students' attitude and interest in computing courses [33]. However, in contrast with this research, those assessments of participants' view of CT were conducted using human raters.

B. Topic Modeling in Education

Topic modeling is a statistical framework that was initially developed as a natural language processing tool to make unsupervised inferences of the latent topics in text collections. A topic model extracts topics or word probability distributions from a text according to a ranking of all distinct words deemed relevant to each topic, resulting in collections of meaningfully-related words. The profile of all the tokens (words) assigned to a topic is examined to understand that topic.

Several approaches to topic modeling have been proposed [34]. First, a linear algebra approach called latent semantic analysis (LSA) was proposed [35], followed by probabilistic approaches, such as the probabilistic latent semantic analysis or pLSA [36]. For instance, the foundational probabilistic topic model Latent Dirichlet Allocation (LDA) [37]-[39] is one of the most popular topic modeling methods building on pLSA. LDA constitutes a generative unsupervised model, using the Dirichlet probability distribution to draw the topics of the topic model.

LDA assumes that each document is generated by a mixture distribution of different topics, and that each topic is a distribution of different tokens (i.e., words). At a higher level, the LDA approach assigns each token a random topic. Then, it iterates through all the tokens and updates their assignment to a topic to minimize the occurrence of tokens across many topics, while preserving the contexts of those tokens that tend to appear together in documents. This method generates a model of topics as a description of the words that are likely to occur together. More specifically, LDA assumes that any document is generated in four major steps: 1) determine the number of words in a document, 2) choose a mixture distribution of different topics, 3) choose a topic for each word based on the mixture distribution, and, finally, 4) choose a word from that topic. For example, suppose there is a document about a cat. LDA will follow these steps: first, it will generate a number of words for a document (e.g., 500); second, it will generate a mixture distribution of topics (e.g., 40% about cats, 40% about dogs, and 20% about birds); third, it will choose a topic for each of the 500 words (e.g., the first word's topic is dog); finally, it will choose a word from each of the 500 topics generated in step three (e.g., for the first word, 'Husky' is sampled from the topic of dog).

In practice, LDA parameters are often estimated using the Gibbs sampling to learn the topic model [40]. As mentioned above, topic assignments are performed randomly. Initially, each word has an equal probability of being associated with any of the determined number of topics. With each iteration, the topic assignments per word are adjusted to reflect the probabilistic model of the data, ultimately converging to specific themes. Recently, LDA has been widely used in educational data mining [34] in the analysis of student discourse [41], and also to identify distinct themes of study and classify students according to course enrollments [42]. There are two major applications of LDA. First, it is used as a topic modeling tool to uncover topics from text data—for example, extracting topics from subtitles of online lectures and ranking the importance of those topics [43]. Results are used to automatically indicate to an interested learner the key concepts that the lectures cover. A second common application of LDA is a feature-extraction tool. After important features (topics) are extracted from text data, they can be used to classify or predict certain outcomes of interest—for example, researchers used LDA to extract important features from online course questions and applied support vector machines to predict the learning outcomes of these questions based on the extracted features [44].

III. Methods

A. Participants and Procedure

Participants were n = 139 pre-service teachers (95 females, 43 males, and 1 undisclosed) out of 219 students enrolled in a section of a large undergraduate introductory educational technology course at a research-intensive university in Western Canada taught by the first author in Winter 2018. The course introduces future K-12 teachers to basic concepts and principles of using technology to transform teaching and learning, as well as issues and trends in educational technology. Participants provided online informed consent at the beginning of the class. Participants' distribution in the program was: 46 in Year 1, 59 in Year 2, 32 in Year 3, 1 in Year 4, and 1 not set (i.e., enrolled in open studies), with 41 being Elementary students, 97 being Secondary students, and 1 not set. On the pre-test administered on the first day of class to collect demographic information, only 17 students reported that they had heard the term "computational thinking" before the study. However, when prompted to define computational thinking, most students described it as using technology to learn. That lack of a precise definition of CT was reminiscent of the control condition in a study that sampled pre-service teachers, where only the treatment group received a CT module intervention [20]. Of these students, only ten viewed CT as a problem-solving systematic thinking process.

The platforms supporting this study were Code.org and Moodle. First, as part of their course assignment, participants spent 20 hours during the first two months of the term individually completing the Accelerated Intro to Computer Science course offered by Code.org, which took students through 98 steps and several levels of learning. At each step, participants were given a block-based visual programming scenario to solve; some scenarios were based on the player's ability to combine actions in a specific order, others required the player to use conditional statements or iterations, or to abstract out a code sequence into a function, so as to move a virtual character from one point to another in a maze-like environment. The students were free to solve the game levels in any preferred order. Second, they were then required to upload an up to 500-word reflection on their experience onto the Moodle-supported course learning management system. Students received a mark only if they submitted the reflection. They were given two prompts to help structure and guide their reflections:

- Reflect on the knowledge, skills, and attitudes that you learned while playing through the Code.org activity. Specifically, what aspects of knowledge, skills, and attitudes were highlighted by the Code.org activity? What were you expecting before you started? What aspects did you find enjoyable, easy, or difficult? How did you feel as you went through the activity?
- Classroom use: How would you use this activity in your own classroom? What would you like your future students to learn from this activity?

B. Data Analyses

Topic modeling was employed to extract the essence of the reflections written by the pre-service teachers about their coding experience, in search of any evidence of CT skills or attitudes. This method addresses the main research question of revealing information about pre-service teachers' knowledge, skills, and attitudes on computational thinking. Instead of assigning an entire reflection to a topic, LDA was used to assign each word in each reflection to a topic, regardless of the order of the words in each reflection. This produces a distribution of topic assignments for each reflection as well as a probabilistic distribution of words for each topic. The reflections were up to 500 words long, but LDA is not sensitive to the length. Thus, the topic model probabilistically associates each reflection with latent topics via the words contained in that reflection. In the analysis, each participant was represented by their reflection. Topic modeling associates participants and their words with the latent topics revealed.

Before topic modeling, participants' reflections were preprocessed in *Python 3.6*. First, all words were changed to lowercase, and punctuation and English stop words that do not carry meaning (e.g., "the", "a", "of") were removed from the text, as they do not contribute to topic modeling and add noise to the topic identification process. Next, each word was lemmatized to its dictionary form (e.g., 'apples' was lemmatized into 'apple') to make the vocabulary list more concise. Rare vocabularies (words used less than twice in the text) were removed from the data because, statistically, they did not contribute to topic analysis. Finally, each reflection was converted into a frequency vector for all the remaining vocabularies. This step is necessary, as each participant is assigned to the topic that appeared most frequently in that participant's reflection. All the data preprocessing procedures were conducted using the Python NLTK library [45].

After data preprocessing, the Latent Dirichlet Allocation (LDA; [37]) unsupervised machine learning technique, using the *Python LDA 1.0.5* library was used to extract topics from students' reflections. In LDA, each topic feature computes the frequency of words in an utterance statement, matching its corresponding topic dictionary. One parameter that needs to be provided to the topic modeling algorithm is an appropriate number of topics. There are no statistical methods that can definitively determine the number of topics, although there are methods for automatic inference of optimal values [46]. Thus, the number of topics was determined according to the ease of interpretation and visualization. Five to ten topics were extracted and the number of topics that led to the best interpretation constituted the final number of topics. The next section displays the most representative sentences for each topic. An abridged version of each topic was extracted from the original data as the most representative sentences in the text, using the *gensim* NLP summarization algorithm [47], which is based on ranks of text sentences using a variation of the similarity function of the TextRank algorithm [48]. One way to summarize a topic model is to extract the words that are most probable in the context of a given topic for a set of representative "topic keys".

TABLE I
SIX TOPICAL FEATURES EXTRACTED FROM A TOPIC MODEL FIT OF STUDENTS' CODING REFLECTIONS

Topic	Top 10 key words
1	Computer, course, science, learning, basic, programming, understanding, computational, logic, believe
2	Code, activity, knowledge, game, video, easy, ability, thinking, student, puzzle
3	Assignment, way, code, think, really, fun, complete, different, difficult, program
4	Level, able, task, complete, basic, quite, order, way, angle, function
5	Level, problem, code, skill, student, solving, thinking, program, able, develop
6	Coding, use, language, task, concept, teaching, taught, student, creative, video

IV. RESULTS

After running LDA iteratively for five to ten topics on students' reflections and interpreting each solution to answer the main research question, six topics emerged that provided an optimal interpretation (that is, adding more topics seemed to produce redundant information). Table I summarizes the topic model based on participants' reflections, in six topics, together with the top ten key words (the most probable words in each topic) in descending order per topic (each topic being a distribution over all words). Participants were assigned to the most frequent topic in their reflection. To interpret each topic, both the key words and the most representative sentences for that topic (i.e., sentences with the highest probability for that topic) were examined.

About 6.5% of the student reflections were assigned to Topic 1, whose key words were related to students' deep reflections about their understanding of the definition of CT. Typical sentences were: "*Computational thinking can be described as the thought process that allows for the analysis of a problem and the logical sequence of steps that can be followed to present a solution. This form of thinking is important for understanding how computers think and work. The levels incorporated in the Code.org course provide you with the task of using logical reasoning to place steps in a specific sequence that allows an objective to be completed.*"

About 10.1% of student reflections were assigned to Topic 2, whose key words were related to the gamification aspect of the Code.org visual programming activity, and to strategies and attitudes of resilience towards problem solving under the challenges encountered while tackling the Code.org learning task. Typical sentences were: "*Introducing coding in a game type format was quite enjoyable, as I feel that had it been in a non-game type delivery, that I would have lost interest and been easily frustrated when I encountered difficulties*". The key words (e.g., "student") suggest that, in their metacognitive reflections, pre-service teachers took the perspectives of their future students. They considered their teaching impact on their own students' CT learning, understanding, problem-solving experiences and perceptions.

About 66.7% of the student reflections were assigned to Topic 3, whose key words were related to Topic 2 and to the meaning-making drawn from the assignment activity. Similar

to Topic 2, this topic highlighted the gaming aspect of the assignment that made the task a positive experience. For instance, students used positive-affect words, such as 'fun'. This topic illustrates CT practices, such as problem decomposition as well as testing and debugging. Typical sentences include: "*This assignment improved my ability to solve puzzles, and to break down an action into very small steps. While dealing with some of the more difficult puzzles, it was important to be patient as it took a few tries to get the puzzle correct.*" It also emphasizes CT perspectives, such as the persistence needed to solve a challenge, as with to Topic 2.

Only 2.2% of the students were assigned to Topic 4, but it is important to note that students' reflections were classified based on the most frequent topic in their writing, which often consisted of several topics. This low percentage does not necessarily indicate the topic was rare, but rather that few students used the topic as the main focus in their writing. The key words of Topic 4 were related to CT concepts (e.g., functions) and to specific game experiences and programming solutions. Sample sentences were: "*The trickiest part of this level is that you had to insert a command of move forward into each pre-set function, which were 'remove stack of 4 piles' and 'fill stack of 2 holes'.*"

About 11.6% of students were assigned to Topic 5, whose key words were related to CT perspectives, such as strategies and attitudes towards problem solving. For instance: "*Often times, if a level seemed too difficult in the moment I would take a break, come back, try again, and eventually solve it.*"

About 2.9% of students were assigned to Topic 6, with key words reflective of their experiences with the programming language. Students applied a constructivist lens by meaningfully relating their experiences to prior knowledge, as in: "*Working through the coding assignment, I realized that it had its own language. A language that was foreign to me and structured differently than any language I know.*"

V. Discussion

This study addressed the main research question of exploring meaningful themes from pre-service teachers' views on computational thinking, and was conducted in an authentic setting, an undergraduate course, which supports its ecological validity. It contributes to understanding the role that technology has in distilling ideas of CT in pre-service teachers. Reflection narratives on their coding experience constitute a first step in understanding pre-service teachers' CT preparedness to support their own students in developing CT skills. Based on their CT definitions on the pre-test compared with their post-activity reflection, findings show that the coding activity helped participants reflect more deeply on the fundamentals of CT and on how prepared they are to use CT in their own teaching practice and subject domains. The pre-test CT definitions were similar to those of the control group in Yadav et al.'s quasi-experimental study [20], but the post-activity reflections matched those of the treatment group in that study. Pre-service teachers reflected on their learning by referring to their previously-acquired knowledge and cultural contexts, which suggests a more meaningful engagement with CT. Results were largely in accord with previous research, showing an overall limited understanding of CT. In a study surveying 134 pre-service teachers, researchers found participants had a cursory understanding of CT [16], [20], [49]. In the future, participants will receive a formal lecture on CT and the coding activity.

In line with the CT frameworks outlined above [3], [30], most of the topics identified (e.g., Topics 1, 3, 4, and 5) refer to learning processes and CT *concepts* (e.g., functions), *practices* (e.g., problem solving, testing and debugging), and *perspectives* (e.g., patience, perseverance). For instance, three of the pillars of CT [29] were salient in students' reflections: algorithmic thinking was exemplified by Topic 1 ("logical sequence of steps that can be followed to present a solution"), while problem decomposition and testing-and-debugging practices were exemplified by Topic 3. This is not surprising, as the Code.org activity was new to the students and they needed to understand and master these skills to perform the task. As the scenarios increased in complexity, students would have had to apply more testing-and-debugging skills to make their code work. Other important cognitive CT tools, such as functions in Topic 4, simulation in Topic 2 (using games as vehicles for CT skills), and meta-cognitive skills in Topic 5 (i.e., perseverance) were highlighted by the analyses.

Many students emphasized in their reflection some fundamental elements of CT. They used CT terminology, showing the incipient stages of developing a "computational-thinking language" to "describe computation, abstraction, and information" [50]. Results from this study echo findings of the treatment group in a study that sampled pre-service teachers, in which the intervention used CT modules to introduce participants to CT in an educational psychology course [20].

In concordance with constructivist learning theories [51], participants related their CT skills to their prior knowledge and experiences (Topic 6) and pointed out challenges (Topics 3 and 4). Importantly, examining pre-service teachers' attitudes towards CT during problem solving may reveal their approaches to learning by themselves and when they teach their own students. The themes extracted by the topic modeling approach will help teachers adapt their instruction to students' CT learning challenges, as they have first-hand experience themselves with CT through the Code.org course. Results will also help researchers to evaluate programs of instruction (for example, teaching CT) to determine how different interventions can aid instructors in teaching CT and students in learning CT. Thus, results can contribute to informing and facilitating the teaching and learning of CT. In turn, they can support the preparation of 21st-century learners, empowering them to solve complex problems and contribute productively and innovatively to an increasingly digital society [52].

This study is relevant for the development of CT skills for pre-service teachers because teachers' knowledge plays an important role in affecting youth's decisions to pursue higher education in Science, Technology, Engineering, and Mathematics (STEM) fields [13], [53], [54]. Notably, the words "enjoyable" emerging in Topic 2 and "fun" emerging in Topic 3 prompt future sentiment analysis research studies. This is particularly encouraging, as research showed that stereotypes

and negative attitudes toward programming continue to deter many from studying computer science and pursuing careers in technology [55]. In addition, Topic 3 highlighted both "fun" and "difficult", which in the game environment can be blended to produce the state of flow necessary to persist in the presence of increasing challenges [56]. Moreover, Topic 2 highlighted a desired outcome of an activity such as Code.org, namely interest in the activity due to its game-based nature. Together, Topics 2 and 3 reveal several positive attitudes of pre-service teachers that may lead to more involvement with CT for themselves and, hopefully, for their future students, especially when CT is accompanied by game-based activities.

Empowering learners to acquire and deepen their knowledge of CT can enable them to innovate in a culture of rapid technological change, especially because CT is considered to be a skill for everyone, not only computer scientists, if they are to thrive in the 21st century digital age [5], [7]. Moreover, CT may help reduce inequality, for example, the wage gender gap [57] and the underrepresentation of women in STEM fields [58], because early computing experiences are crucial for cognitive and social development as well as for future participation in STEM fields [13]. Finally, the computational thinking reflective narrative of pre-service teachers has yet to be examined using topic modeling.

A. Limitations

One limitation of this study stems from the method used to analyze the data. It is possible that different topics would emerge if the same analysis were performed on a different dataset, and that the results have limited generalizability. Although it has been found recently that performance of topic models is inconsistent across a wide variety of real-world corpora [24], the present dataset consisted of text reflections of up to 500 words on a specific experience of taking a block-based visual programming course. Student reflections on this experience are thus likely to be similar, given the authenticity of the themes extracted through topic modeling, which were in concordance with similar attitudes emerging from the computational thinking literature. Second, although topic modeling assigns students to the most frequent topic in their reflection and describes themes that are statistically distinguishable, the topics do not necessarily characterize individual students, who may display a combination of complex behaviors. Thus, interpretations of the results must be done with caution. Third, students received either a complete or incomplete mark for submitting their reflections or not. Thus, no comparative qualitative analysis was conducted by a human rater. Future studies will compare the topics identified by humans to those automatically extracted using topic modeling. Fourth, participants' attitudes were not measured before and after the coding intervention. In a follow-up study, a pre-test and a post-test of attitudes towards CT will be administered. For instance, in the case of in-service teachers, prior research recorded significant improvements in their views on computing science even after a weekend-long CT workshop [21]. Finally, participants were in different years of their program, mostly in years 1 to 3, but all were novices in computing, as is the case with the majority of pre-service teachers.

B. Implications

The insights gained from the results of this study have practical relevance and ramifications for the design of university education courses on CT for pre-service teachers. It is apparent from pre-service teachers' reflections that they are not only interested in CT skills and in attitudes surrounding CT, but are also interested in the pedagogical aspects of teaching CT, such as in a step-by-step manner as suggested by a participant's reflection highlighted by Topic 1, and gamification-based as suggested by a participant's reflection highlighted by Topic 2. Their answers reveal pre-service teachers' unease with their current preparation for teaching CT, given that CT concepts that are not only challenging to understand in themselves, without having dedicated CT courses in their training programs, but are also challenging to teach to their future students. Further, they manifested an interest in strategies to learn CT, such as exercising patience, decomposing a complex problem into simpler subproblems as suggested by Topics 3 and 5, and revisiting difficult problems as suggested by Topic 5. Universities can benefit from conducting surveys with pre-service teachers and listening to their points of view on CT when developing post-secondary courses, programs of instruction, and assessments. For example, instructors may combine important courses, such as the present Code.org course, with modules on pedagogy and on applying CT in different subjects. This study supports the introduction of computational thinking to pre-service teachers as part of an undergraduate introductory educational technology course. Further supporting these results, related research urged training for pre-service teachers to be better prepared to teach CT in several K-12 subjects [16]. It is believed that appropriate guidance from teachers can enable students to learn to apply CT strategies on their own [59].

C. Future Work

Based on the results of the current study, future research can use topic modeling to extract themes from the course's forum posts to provide a deeper understanding of students' topics of interest in each thread. For instance, the themes extracted from these analyses can be used to generate feedback tailored to students' needs as well as to identify students' strengths and weaknesses. Moreover, topic modeling can provide a complementary tool that empowers instructors to analyze students' written output. The topic model that fits the data can identify commonalities and frequent themes that could be missed by a manual analysis of text. The novel topics extracted could be used as the starting point for further analyses or as a point of focus. Specifically, a topic model can prompt an instructor to review examples of student reflections closely related to a topic, together with the top words associated with that topic, to obtain a more in-depth representation of that topic. Future research will focus on pre-service teachers' preferences for specific topics and on any perceptible change in topics with changes in the content taught.

In future studies, sentiment analysis will be conducted to

further explore pre-service teachers' attitudes towards computational thinking. Human raters will examine each student reflection and extract themes based on salient CT concepts, practices, and perspectives. These themes will be compared with the LDA-generated topical features. Finally, students' narratives may reveal information about their learning outcomes. In a recent study, students' language use in a MOOC helped differentiate lower from higher performers [60]. Thus, future studies will explore the relationship between topics generated using topic modeling and learning outcomes. Because the field still lacks a systematic way of understanding the effect of different topic modeling algorithms on the performance of the topic models, implementations of LDA other than Gibbs sampling, and methods other than LDA, will be applied to this dataset; the resulting models will be compared and contrasted. The introduction of CT in non-education undergraduate courses will also be explored, as prior research found that CT in undergraduate science courses improved students' attitude and interest in taking computing courses [33]. Based on the present research, it would also be important to ascertain pre-service teachers' careers for evidence of increased use of computing skills (e.g., how many participants teach computing science courses or use CT principles in other subjects). Also, CT knowledge is thought to aid problem-solving skills in other disciplines [9], [15], so a study that tests this hypothesis focusing on pre-service students' problem-solving abilities in other courses would be valuable. It would be helpful to know whether participants continued to explore the Code.org website in search for other challenges or whether they manifested interest in other types of CT endeavors. Finally, a follow-up study will examine pre-service teachers' ability to transfer their CT knowledge, skills, and attitudes to other domains.

VI. Conclusion

The study analyzed pre-service teachers' written reflections on their views on CT after an online programming course. Findings identified separable themes or topics of interest in pre-service teachers' coding reflections. Results of employing topic modeling, a machine learning modeling technique, to extract topical features from these reflections showed that the coding activity aided pre-service teachers' understanding of CT. It also showed that participants envisioned integrating CT into their own teaching through algorithmic thinking, abstraction, problem decomposition, and problem solving, beyond the use of computers. Pre-service teachers used CT terms (concepts, practices, and perspectives) and drew many connections between the coding activity and their prior knowledge and experiences. Thus, the model was useful in identifying CT-related reflections in an undergraduate educational technology introductory course. It contributed to deepening and advancing understandings of CT in a population of pre-service teachers, reinforcing the need for ways to incorporate innovative pedagogy aligned with 21st-century competencies. These results constitute a starting point in shaping the role of pre-service teachers in teaching CT and helping post-secondary instructors create differential courses for pre-service teachers that are customized to the knowledge and attitudes of their students towards CT. The findings are important in devising evidence-based teacher-preparation curricula aligned with pre-service teachers' interests.

Acknowledgment

The authors would like to thank all the students who participated in this study and the funding agencies that supported this research: the Social Sciences and Humanities Research Council of Canada - Insight Development Grant (SSHRC IDG) RES0034954, the Natural Sciences and Engineering Research Council (NSERC DG), and the Killam Cornerstone Operating Grant RES0043207.

References

[1] B. Czerkawski (2015). "Computational thinking in virtual learning environments". In *Proc. of E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*, Association for the Advancement of Computing in Education (AACE), Chesapeake, VA, 2015, 993-997.

[2] S. Grover (2015). "Systems of Assessment" for Deeper Learning of Computational Thinking in K-12. *AERA*.

[3] S. Grover and R. Pea (2018). "Computational Thinking. A competency whose time has come". *Computer Science Education: Perspectives on Teaching and Learning in School*, 19.

[4] S. Papert (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books.

[5] J. M. Wing (2006). "Computational thinking". *Communications of the ACM, 49(3),* 33–36.

[6] J. M. Wing (2008). "Computational thinking and thinking about computing". *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 366(1881)*, 3717-3725.

[7] J. M. Wing (2011). "Research notebook: Computational thinking – What and why?" *The Link Magazine*, Spring. Carnegie Mellon University, Pittsburgh, PA. Retrieved from http://link.cs.cmu.edu/article.php?a=600

[8] J. M. Wing (2016). "Computational thinking, 10 years later". *Communications of the ACM, 59(7)*. Retrieved from: https://cacm.ac.org/blogs-cacm.

[9] M. Guzdial (2008). "Education paving the way for computational thinking". *Communications of the ACM, 51*(8), 25-27.

[10] K-12 Computer Science Framework (2016). Retrieved from https://k12cs.org.

[11] ISTE (2011). "Computational Thinking: leadership toolkit". Retrieved from: https://c.ymcdn.com/sites/www.csteachers.org/resource/resmgr/471.11CTLeadershiptToolkit-S.pdf.

[12] G. H. Fletcher and J. J. Lu (2009). "Education human computer skills: rethinking the K-12 experience". *Communications of the ACM, 52*(2), 23-25.

[13] R. Tai, C. Q. Liu, A. V. Maltese, and X. Fan (2006). "Career choice: Enhanced: Planning early for careers in science". *Science, 312*, 1143-1144.

[14] C. Corbett and C. Hill (2015). "Solving the equation: The variables for women's success in engineering and computing". *The American Association of University Women*, Washington, DC.

[15] V. Barr and C. Stephenson (2011). "Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community?" *ACM Inroads, 2*(1), 48–54.

[16] A. Yadav, S. Gretter, J. Good, and T McLean (2017). "Computational thinking in teacher education." In *Emerging research, practice, and policy on computational thinking*. 205-220. Springer, Cham.

[17] S. Y. Lye and J. H. L. Koh (2014). "Review on teaching and learning of computational thinking through programming: What is next for K-12?". *Computers in Human Behavior, 41*, 51-61.

[18] E. Prieto-Rodriguez and R. Berretta (2014). "Digital technology teachers' perceptions of computer science: It is not all about programming." In *Proc. of the IEEE Frontiers in Education Conference (FIE)*. 1-5.

[19] A. Yadav, N. Zhou, C. Mayfield, S. Hambrusch, and J. T. Korb (2011). "Introducing computational thinking in education courses." In *Proc. of the 42nd ACM Technical Symposium on Computer Science Education*.

[20] A. Yadav, C. Mayfield, N. Zhou, S. Hambrusch, and J. T. Korb (2014). "Computational thinking in elementary and secondary teacher education." *ACM Transactions on Computing Education (TOCE), 14*(1), 5.

[21] L. Blum and T. J. Cortina (2007). "CS4HS: An outreach program for high school CS teachers." In *Proc. of the 38th SIGCSE Technical Symposium on Computer Science Education (SIGCSE'07)*.

[22] P. Morreale and D. Joiner (2011). "Changing perceptions of computer science and computational thinking among high school teachers." *J. Comput. Sci. Colleges 26*(6), 71–77.

[23] S. Grover (2017). "Assessing algorithmic and computational thinking in K-12: Lessons from a middle school classroom". In *Emerging research, practice, and policy on computational thinking*. 269-288. Springer, Cham.

[24] H. Shi, M. Gerlach, I. Diersen, D. Downey, and L. A. Amaral (2019). "A new evaluation framework for topic modeling algorithms based on synthetic corpora." arXiv preprint arXiv:1901.09848.

[25] P. J. Denning and P. A. Freeman (2009). "The profession of IT: Computing's paradigm." *Communications of the ACM, 52*(12), 28–30.

[26] S. Goldstein, D. Princiotta, and J. A. Naglieri (2015). *Handbook of intelligence*. Evolutionary Theory, Historical Perspective, and Current Concepts. New York, NY: Springer.

[27] J. Cuny, L. Snyder, and J. M. Wing (2010). "Demystifying computational thinking for non-computer scientists". Retrieved from: http://Www.Cs.Cmu.Edu/~ CompThink/Resources/TheLinkWing.Pdf.

[28] S. Grover and R. Pea (2013). "Computational thinking in K–12: A review of the state of the field". *Educational researcher, 42*(1), 38-43.

[29] V. J. Shute, C. Sun, and J. Asbell-Clarke (2017). "Demystifying computational thinking". *Educational Research Review, 22*, 142-158.

[30] K. Brennan and M. Resnick (2012). "New frameworks for studying and assessing the development of computational thinking". Paper presented at the *Proc. of the 2012 AERA*, Vancouver, Canada.

[31] V. Allan, V. Barr, D. Brylow, and S. Hambrusch. 2010. "Computational thinking in high school courses". In *Proc. of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE'10)*.

[32] D. D. Garcia, C. M. Lewis, J. P. Dougherty, and M. C. Jadud (2010). "You might be a computational thinker!". In *Proc. of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE'10)*.

[33] S. Hambrusch, C. Hoffmann, J. T. Korb, M. Haugan, and A. L. Hosking (2009). "A multidisciplinary approach towards computational thinking for science majors". *ACM SIGCSE Bulletin, 41*(1), 183-187.

[34] J. Boyd-Graber, Y. Hu, and D. Mimno (2017). "Applications of topic models". *Foundations and Trends in Information Retrieval*, 11(2-3), 143-296.

[35] S. Deerwester, S. Dumais, T. Landauer, G. Furnas, and R. Harshman (1990). "Indexing by latent semantic analysis". *Journal of the American Society of Information Science, 41*(6). 391–407.

[36] T. Hofmann (1999). "Probabilistic latent semantic analysis". In *Proc. of Uncertainty in Artificial Intelligence*.

[37] D. M. Blei, A. Y. Ng, and M. I. Jordan. (2003). "Latent Dirichlet allocation". *Journal of Machine Learning Research, 3,* 993-1022.

[38] D. M. Blei (2012). "Probabilistic topic models". *Communications of the ACM, 55*(4), 77-84.

[39] S. P. Crain, K. Zhou, S. H. Yang, and H. Zha (2012). "Dimensionality reduction and topic modeling: From Latent Semantic Indexing to Latent Dirichlet allocation and beyond". In *Mining Text Data*. 1-522. Springer.

[40] C. H. Cheng and W. L. Hung (2018, June). "Tea in Benefits of Health: A Literature Analysis Using Text Mining and Latent Dirichlet Allocation". In *Proc. of the 2nd International Conference on Medical and Health Informatics*. 148-155. ACM.

[41] A. Ezen-Can, K. E. Boyer, S. Kellogg, and S. Booth (2015). "Unsupervised modeling for understanding MOOC discussion forums: a learning analytics approach". In *Proc. of the Fifth International Conference on Learning Analytics and Knowledge (LAK '15)*. ACM, New York, NY, USA, 146-150.

[42] B. Motz, T. Busey, M. Rickert, and D. Landy (2018). "Finding Topics in Enrollment Data". In *Proc. of Educational Data Mining*.

[43] J. Zhu, X. Li, Z. Wang, and M. Zhang (2017). "An effective framework for automatically generating and ranking topics in MOOC videos". In *Proc. of Educational Data Mining*. Wuhan, China.

[44] S. Supraja, K. Hartman, S. Tatinati, and A.W.H. Khong (2017). "Toward the automatic labeling of course questions for ensuring their alignment with learning outcomes". In *Proc. of EDM*. Wuhan, China.

[45] S. Bird, E. Klein, and E. Loper (2009). *Natural language processing with Python: analyzing text with the natural language toolkit.* O'Reilly Media, Inc.

[46] R. Arun, V. Suresh, C. V. Madhavan, and M. N. Murthy (2010). "On finding the natural number of topics with Latent Dirichlet allocation: Some observations". In *Advances in Knowledge Discovery and Data Mining*, Springer, Berlin. 391-402.

[47] Gensim Summarization Algorithm (2018). https://radimrehurek.com/gensim/summarization/summariser.html

[48] F. Barrios, F. López, L. Argerich, and R. Wachenchauzer (2016). "Variations of the Similarity Function of TextRank for Automated Summarization". https://arxiv.org/abs/1602.03606

[49] J. Good, A. Yadav, and A. Lishinski (2016). "Measuring computational thinking preconceptions: analysis of a survey for pre-service teachers' conceptions of computational thinking". In *Society for Information Technology and Teacher Education (SITE)*. Savannah, GA.

[50] J. J. Lu and G. H. Fletcher (2009). "Thinking about computational thinking". In *ACM SIGCSE Bulletin, 41*(1). 260-264. ACM.

[51] J. Piaget (1964). In Ripple R. E., Rockcastle V. N. (Eds.), Quoted by Eleanor Duckworth in "Piaget rediscovered. A Report of the Conference on Cognitive Studies and Curriculum Development". *Cognitive Studies and Curriculum Development Conference* (1st ed.). Cornell University, Ithaca: ERIC.

[52] C. C. Selby (2015). "Relationships: computational thinking, pedagogy of programming, and Bloom's Taxonomy". In *Proc. of the Workshop in primary and secondary computing education*. 80-87. ACM.

[53] J. Ehrlinger and D. Dunning (2003). "How chronic self-views influence (and potentially mislead) estimates of performance". *Journal of Personality and Social Psychology, 84(1)*.

[54] J. Margolis and A. Fisher (2002). *Unlocking the clubhouse: Women in computing*. Cambridge, MA: MIT Press.

[55] P. Charters, M. J. Lee, A. J. Ko, and D. Loksa (2014). "Challenging stereotypes and changing attitudes: the effect of a brief programming encounter on adults' attitudes toward programming". In *Proc. of the 45th ACM technical symposium on CS education*. 653-658. ACM.

[56] M. Csikszentmihalyi (1997). *Finding flow: The psychology of engagement with everyday life*. Basic Books.

[57] C. Ashcraft, E. Eger, and M. Friend (2012). *Girls in IT: The Facts*. Boulder, CO: National Center for Women & Information Technology (NCWIT).

[58] Pew Research Center (2018). "Women and Men in STEM Often at Odds Over Workplace Equity". Retrieved from http://www.pewsocialtrends.org/2018/01/09/women-and-men-in-stem-often-at-odds-over-workplace-equity.

[59] NRC (2010). "National Research Council. Report of a workshop on the scope and nature of computational thinking". *National Academies Press*.

[60] N. M. Dowell, O. Skrypnyk, S. Joksimovic, A. C. Graesser, S. Dawson, D. Gaševic, ... and V. Kovanovic (2015). "Modeling Learners' Social Centrality and Performance through Language and Discourse". *International Educational Data Mining Society*.