Analyzing the Effectiveness of Reflection Prompts Accompanying Cybersecurity Assignments Using Natural Language Processing

Cheryl Resch
Engineering Education Department
University of Florida
Gainesville, FL
cheryl.resch@ufl.edu

Christina Gardner-McCune
Computer and Information Science and Engineering
University of Florida
Gainesville, FL
gmccune@ufl.edu

***Abstract*—This research paper describes the use of natural language processing to categorize reflection responses according to their level of learning. This could help instructors in assessing the effectiveness of the prompts in encouraging meaningful reflection and students' engagement with the problem. Reflective practice is the process of using one's beliefs and prior experiences to analyze a problem; it is making meaning from experience. Answering reflection prompts has been shown to aid students in learning problem solving skills. This paper describes results of an experiment in which a pretrained bi-directional encoder representations for transformers (BERT) model was used to classify responses to reflection responses. In an earlier experiment, a method was developed for deductively coding prompts using four progressive levels of learning derived from Dewey and Moon: Noticing, Making Sense, Making Meaning, and Transformative Learning. Responses to reflection prompts designed to encourage different levels of learning were manually coded. In the experiment described in this paper, manually coded responses were used to fine tune models that classify reflection prompts for the presence of each level of learning. The models performed well on the test set, indicating substantial agreement with the assigned manually codes. The models were then used to classify responses to reflection prompts accompanying assignments in on input validation vulnerabilities in two Computer Science courses. Three reflection prompts meant to encourage successive levels of learning were given with the assignments. The models proved to be effective in classifying responses, and gave insight into the effectiveness of the reflection prompts, and gave insight into students' engagement with the material in two different classes.**

***Index Terms*—Reflective practice, Undergraduate assessment tools**

I. Introduction

Undergraduate Computer Science education entails more than imparting facts and learning to use tools. The Association for Computing Machinery (ACM) Computer Science Curricula 2013 states that "Graduates need to understand how to apply the knowledge they have gained to solve real problems, not just write code and move bits" [1]. Thus, computing educators must equip their students with skills for solving problems. Dewey [2] defines reflection as a way of thinking that results from examining beliefs when habitual thinking is not adequate to solve a problem. Dewey evokes the scientific method of observation, formulation and testing of hypotheses and suggests that reflective thinking must occur in order to hypothesize new ways to think about and solve a problem. Reflection prompts can guide students to articulate a problem and develop their problem-solving skills [2]. Assessing students' responses to reflection prompts to determine the effectiveness of the prompt and the depth of learning demonstrated in the response is labor intensive. This makes assigning reflection prompts in larger classes prohibitive. This paper explores the use of natural language processing (NLP) to analyze the content of reflection responses.

II. Theoretical Background

Schon's [3] "Reflective Practitioner" advocated for an expansion of the education of professionals beyond imparting a collection of facts and skills. Schon suggested that professionals must become familiar with the process of solving problems that they have not seen before, which requires them to engage in "Reflection-in-action" [3]. In the context of learning and problem solving, when a learner has a problem that cannot be solved in the habitual way, reflective thinking must occur in order to hypothesize new ways to think about and solve the problem. "Demand for the solution of a perplexity is the steadying and guiding factor in the entire process of reflection" [2]. Dewey defined distinct steps in reflection:

- The occurrence of a difficulty
- Definition of the difficulty
- Suggestion of a possible solution
- Rational elaboration of an idea
- Corroboration of an idea and formation of a concluding belief

Moon [4] extends the concepts of reflection from "difficulties," or solving specific problems in professional practice, to the process of learning in general in undergraduate education. Moon describes the stages of learning as a continuum of surface learning to deep learning that correspond to Dewey's stages of reflection:

- Noticing - naming the problem and imparting facts
- Making Sense - describing the problem, but not in relation to previous understandings

- Making Meaning - the beginning of a holistic view
- Working with Meaning - integrating ideas
- Transformational Learning - new learning has transformed understanding, there is a restructuring of ideas

Requiring students to memorize facts only gets them to the first stage of learning. The ultimate goal is for students to update the models in their minds so that they can use the concepts they have learned to solve future problems. Thus, Moon expands the idea of the reflective process from solving a problem or responding to a "difficulty" to making connections and developing a holistic view so students have a base of knowledge to draw on in new situations [4]. Instructors can guide learners through these stages of learning with reflection prompts [5] [6]. This paper describes the use of NLP to examine responses to general open-ended reflection prompts to posit which stage of learning the students are in.

Rogers [5] reviewed the writings of Dewey and Schon and developed a set of requirements relating to the process of reflection for teaching and learning in higher education. Rogers points out that to enable learners to engage in reflection, they should be prompted to think about an unusual or perplexing situation. In the case of a computer science (CS) undergraduate, this could be in the context of writing an algorithm to solve a problem or working with a group to figure out how to meet software engineering requirements. Rogers suggested several frameworks for guiding learners through reflection. One of these frameworks uses a set of structured questions about a particular experience, and is the one we use in our study. The work of Schon, Dewey, Moon, and Rogers form our theoretical foundation and we used their ideas as a lens to interpret student reflections.

III. Previous Work

A. Reflection Prompts in CS Education

In a review of the CS education literature, we found that reflection has been suggested as a way of enhancing students' problem solving skills and self-awareness of their software development process. Several papers pointed out that students find reflection difficult and suggest ways to scaffold the process. Coffey and Owsnkicky [7] [8] and Nylen and Issomottonen [9] reported on adding reflective writing to software engineering courses and discussed how to encourage students to thoughtfully participate in the activity, such as including length requirements. Caruso et. al [10] reported on having first year CS students reflect on how they could improve their software development process. They found that the plans students created were often vague and that they need scaffolding for developing reflection skills. Kakavouli and Metaxas [11] experimented with reflection questionnaires in a CS2 class and suggest that instructors provide guided reflection prompts, such as "What did you learn?" In this way, students can take charge of their own learning and the instructor can scaffold the process with guided prompts.

Several CS education papers explored the use of guided reflection prompts to promote deeper conceptual learning and retention of course content in introductory programming courses [12], Computer Architecture, and Distributed Computing [13]. Pears and Larzon [13] describe a qualitative analysis of reflections collected from 42 students in a Computer Architecture course and 22 students in a Distributed Computing course. Students wrote a reflection about course modules using prompts such as the following: "Was there something that was especially interesting? If so: what and why?". When students were asked how they benefited from the reflection assignments, the dominant answer was that "they perceive that they have learned more by having to think about what they heard during the lecture and what they really did not understand that well" [13]. The authors concluded that the reflection prompts "encourage behaviours that have been linked to deep learning" [13].

Alzaid and Hsiao [12] describe two studies of reflection in an introductory Java course and also stress the importance of guided prompts. Students reflected on answers to optional daily quizzes to self-assess and monitor their learning progress. In their first study, students were asked to reflect on answers to quiz questions, but without any prompts to guide them. In the second study, students were given the prompt: "Can you tell us why you chose this answer?" Alzaid and Hsiao found that giving students a specific prompt resulted in more constructive reflections and improved performance on quizzes [12].

Our previous work includes an experiment comparing the performance on exams for students who responded to reflection prompts with that of students who did not not [14]. Students were assigned to one of two groups. Prompts designed encourage students to move along the stages of learning as defined by Moon were assigned to one of the groups while the other group did not have to respond to reflection prompts. The study found that students who answered reflection prompts performed better on exams. This work uses similar prompts with assignments on input validation in a CS1 and in a Computer Organization course.

B. Automated Assessment of Responses

Given the prompts developed in previous work, we next explore using NLP for categorizing responses to reflection prompts. Barthakur et al. [15] performed a similar experiment in which machine learning was used to classify responses of professionals to guided reflection prompts in a workplace learning situation into one of four stages of learning. Barthakur et al. use part of speech tagging, word count, sentiment analysis, n-grams and the LIWC (Linguistic Inquiry and Word Count) [16] and Coh-Metrix [17] tools to create a feature set. They then used multiple machine learning algorithms to train models to predict the level of learning in a response, including Logistic Regression, Decision Trees, SVM, and Random Forests. Their models produced Cohen's Kappa [18] values from 0.48 (moderate agreement) to 0.66 (substantial agreement).

Wulff et. al in 2023 [19] demonstrated the effectiveness of fine-tuning the BERT model for classifying reflection responses. Wulff et. al classified reflections of pre-service

Physics teachers using a reflection model [19] that looks for the presence of circumstance, description, evaluation, alternatives, and consequences in responses. They created sentence embeddings using a pre-trained BERT transformer, then used a fine-tuning scheme to create a classifier. They created five models, one each for predicting the presence of circumstance, description, evaluation, alternatives, and consequences. They reported Cohen's Kappa [18] values from 0.66 to 0.75, indicating substantial agreement. In this paper we use a similar strategy for creating models for classifying reflection responses.

IV. Research Questions

In light of the previous work creating a classification scheme for responses to reflection prompts, and demonstration of the effectiveness of reflection prompts in helping students learn, this work explores the use of NLP to classify responses. We seek to answer the following research questions.

- Can NLP be used to accurately classify responses to reflection prompts?
- Can results from NLP models be used to assess the effectiveness of reflection prompts?

V. Methods

In previous work [20], 219 students in a Fall 2019 Data Structures course completed two programming assignments and were asked to respond to reflection prompts after each. This resulted in 1054 responses, which were deductively coded using progressive levels of learning derived from Dewey and Moon: *Noticing, Making Sense, Making Meaning*, and *Transformative Learning*. Table I shows the details of the coding scheme. We combined Moon's *Making Meaning* and *Working with Meaning* into one level that is characterized by the expression of a succession of integrated ideas.

In two rounds of coding and refinement of category descriptions, inter-rater reliability among three raters measured with Cohen's Kappa [18] was found to be 0.83, which is classified as "almost perfect agreement". The 1054 responses each answered the question and imparted facts. No empty or nonsense answers were included. Every response was classified as demonstrating *Noticing*. Table II shows the training set sizes for each category.

Supplementing the training set: The number of positive instances of *Transformative Learning* is very small. *Making Sense* is also quite unbalanced with four times as many positive examples as negative examples. Experiments by Ganesan et. al [21] suggest that 100 samples of each classification is adequate for BERT fine-tuning of a problem of this type. Because of this, the training set was supplemented with responses to reflection prompts given with an assignment on buffer overflow given to Computer Organization students in Fall 2022 to obtain a balanced training set with at least 100 samples of each classification for each of the levels. The assignment was accompanied by the following three reflection prompts:

- What was the most surprising thing you learned from this assignment?

TABLE I
Description of qualitative codes

Level/ Code	Description of code from Moon [4]	Extended Description	Sample Student Response
1 Noticing	Naming the problem and imparting facts		The hardest part of this assignment was understanding how recursion works when used to traverse a tree.
2 Making Sense	Describing the problem, but not in relation to previous understandings	Descriptive words, e.g. fun, tricky, important, annoying, interesting. Description of one part of implementation.	The most difficulty I had was finding a few isolated bugs spread out over the recursive methods. Finding the bugs required *diligent* use of the debugger, as well as *careful* attention to where in the tree the program was at any given time: non-linear data structures and algorithms complicate this process significantly.
3 Making Meaning / Working with Meaning	Integrating ideas, the beginning of a holistic view	Description of problem solving including more than one part of implementation, describes before and after phases of solving the problem, compares two solutions.	*Initially*, I wanted to do an adjacency list graph implementation; however, *I had to convert* to using an adjacency matrix in order to optimize the Dijkstra's algorithm I implemented. ... my Dijkstra's implementation using the adjacency list graph implementation was inefficient with a time complexity greater than O(V cubed) due to non-constant time for isEdge and getWeight. Therefore, I converted my graph implementation to an adjacency matrix in order to get the methods isEdge and getWeight which are used in my Dijkstra's implementation to O(1).
4 Transformative Learning	New learning has transformed understanding, there is a restructuring of ideas.	Plans for the future, lessons to be used in the future, theoretical discussion.	I learned that recursion is an effective tool for solving problems with trees. Prior to this assignment I knew of recursion and how it works, but never had known an effective way to use them or why I would ever want to use them. In this scenario *recursion finally seems to be a useful tool that I can use to solve problems that I may come across in programming*.

TABLE II
Training set sizes for binary classifiers

Level of Learning	N Positive (Class=1)	N Negative (Class=0)
Making Sense	860	194
Making Meaning	441	613
Transformative Learning	52	1002

TABLE III
TRAINING SET SIZES FOR BINARY CLASSIFIERS

Level of Learning	**Positive Training and Validation Set**	**Negative Training and Validation Set**
Making Sense	194 randomly chosen from 2019 dataset+78 randomly chosen from 2022 dataset	194 from 2019 data set +78 from 2022 data set = 272
Making Meaning	441 from 2019 data set +182 from 2022 data set= 623	613 from 2019 data set + 118 from 2022 data set= 731
Transformational Learning	52 from 2019 data set + 51 from 2022 data set =103	52 randomly chosen from each dataset, 104 total

- What are the consequences of a buffer overflow vulnerability?
- What can you do to reduce the likelihood that you will write code with a buffer overflow vulnerability?

463 students responded to these reflection prompts in Fall 2022. Irrelevant, nonsense, or blank responses were excluded. 100 randomly chosen sets of responses to the three prompts, for a total of 300 responses, were coded by the first author and an assistant. Each response was coded for the presence of *Making Sense*, *Making Meaning*, and *Transformative Learning*.

For *Making Sense*, the Cohen's kappa for the 300 responses for the two coders (first author and assistant) was 0.82, or almost perfect agreement. For *Making Meaning*, the Cohen's Kappa for the 300 responses for the two coders was 0.79, or substantial agreement. For *Transformative Learning*, the Cohen's Kappa for the 300 responses for the two coders was 0.96, or almost perfect agreement.

The data are summarized in Table III. For *Making Sense* the negative data set consisted of 194 responses from the 2019 data set plus 78 from the 2022 data set. To make the training set balanced, 194 positive responses were randomly chosen from the 2019 data set and 78 positive responses were randomly chosen from the 2022 data set. The *Making Meaning* data set consisted of 441 positive responses from the 2019 data set and 182 positive responses from the 2019, for a total of 623 positive responses. The negative data set consisted of 613 from the 2019 data set and 118 from the 2019 data set for 731 negative responses. The *Transformational Learning* positive data set consisted of 52 from the 2019 data set and 51 from the 2022 data set. To create a balanced training set, 52 negative responses were randomly chosen from the 2019 data set and 52 responses were randomly chosen from the 2022 data set. The minimum number of positive responses were obtained for the *Transformative Learning* and larger balanced training and test sets were obtained for *Making Sense* and *Making Meaning*.

A. Model

The coded reflections were passed to the HuggingFace [22] pre-trained BERT model to create sentence embeddings. The sentence embeddings were split into a training set, a validation set, and a test set. 15% of the responses for each level were set aside to serve as the test set, and 15% for the validation set. A 70:30 split for training and testing data is common in machine learning algorithms with at least 100 samples, as this ratio tends to avoid over-fitting while leaving enough samples for training [23].

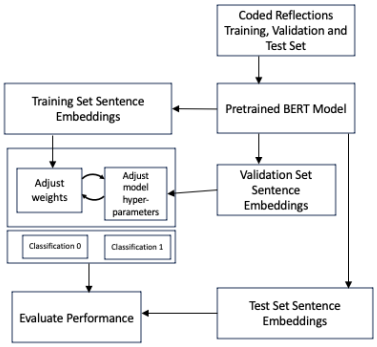

Fig. 1. Fine tuning procedure

TABLE IV
RESULTS FOR MAKING SENSE TEST SET

	Predicted Class →	**0 - Negative**	**1 - Positive**	**Total**
Actual Class ↓				
0 - Negative		29	4	33
1 - Positive		9	36	45
Total		38	40	78

Cohen's $\kappa = 0.67$

Algorithms from the Hugging Face Transformers Python library were used to carry out the fine tuning process. The model was fine-tuned in three epochs. In each epoch, the model goes through training and validation. During training, the training set is input to the model and model weights are adjusted. Next the validation set was used to evaluate performance and adjust hyperparameters such as learning rate. After fine tuning, the test set was used to evaluate the accuracy of the model. Figure 1 shows the schematic for the training procedure.

The dataset for each level of learning was used to create a fine-tuned BERT classifier with an output layer with two class labels: a 1 if the level of learning is present (positive), and 0 if it is not (negative). Table IV shows the confusion matrix for the test set for *Making Sense*. A response is categorized as 0 (negative) if the 0 output is higher than the 1 output, and 1 (positive) if the 1 output is higher than the 0 output. The classifier made a correct prediction for 29 of 33 0 (negative) test set samples, and a correct prediction for 36 of 45 1 (positive) test set samples. This translates to a Cohen's Kappa of 0.67, corresponding to substantial agreement.

Table V shows the confusion matrix for the test set for *Making Meaning*. The classifier made a correct prediction for 91 of 109 positive (1) test set samples, and a correct prediction for 78 of 91 negative (0) test set samples. Cohen's Kappa is 0.69, corresponding to substantial agreement.

TABLE V
RESULTS FOR MAKING MEANING TEST SET

	Predicted Class →	**0 - Negative**	**1 - Positive**	**Total**
Actual Class ↓				
0 - Negative		91	18	109
1 - Positive		13	78	91
Total		104	96	200

Cohen's $\kappa = 0.69$

TABLE VI
RESULTS FOR TRANSFORMATIVE LEARNING TEST SET

	Predicted Class →	**0 - Negative**	**1 - Positive**	**Total**
Actual Class ↓				
0 - Negative		17	1	18
1 - Positive		2	10	12
Total		19	11	30

Cohen's $\kappa = 0.79$

Table VI shows the confusion matrix for the test set for *Transformative Learning*. The classifier made a correct prediction for 10 of 12 positive (1) test set samples, and a correct prediction for 17 of 18 negative (0) test set samples. Cohen's Kappa is 0.79, corresponding to substantial agreement.

These Cohen's Kappa values are similar to those reported by Wulff et al. [19]. The values indicating substantial agreement between human coders and the NLP models and performance similar to previous results provides confidence that it is reasonable to use the NLP models to provide an estimate of the distribution of level of learning indicated in responses to reflection prompts.

VI. STUDY CONTEXT

Over the course of three semesters, Fall 2022, Spring 2023, and Fall 2023, students in two courses were given assignments accompanied with the three reflection prompts given above.

Students in an introductory programming course (hereafter referred to as CS1) had an assignment in which they were introduced to input validation, practiced input validation on some simple Python code, and were shown examples of infamous input validation vulnerabilities, such as Heartbleed. Students in a Computer Organization course had an assignment in which they ran code vulnerable to buffer overflow in a debugger, and looked at the content of stack memory after providing input that is larger than the space allocated for it [24]. Students were then guided through running the vulnerable program and providing input such that execution jumps to a function that was never called, demonstrating for students the true danger of buffer overflow vulnerabilities. The students' responses to the reflection prompts after both assignments were input to the classifiers to assess the presence of *Making Sense*, *Making Meaning*, and *Transformative Learning*.

VII. RESULTS

Over the three semesters, 1387 students completed the input validation assignment and responded to the reflection prompts. Each response was assessed using the classifiers for the presence of *Making Sense*, *Making Meaning*, and *Transformative Learning*. Tables VII-IX show sample responses at each level of learning for the each prompt and Table X shows the fraction of responses at each level of learning. These responses show examples where the classifier correctly classified the presence of descriptive language, putting together ideas, and new learning. A random check indicates that the classifiers coded responses according to the model given in Table I. Since there were approximately 4000 responses, they were not all manually coded, so we cannot compare the NLP performance to ground truth. However, the random check was encouraging and indicated good performance by the classifiers.

42.6% of the responses to "What was the most surprising thing you learned from this assignment" demonstrated *Making Sense*, in which the problem is described. For example, cross-site scripting is described in the example response in Table VII as "common" and "easily...performed". This prompt, which was developed to encourage *Making Sense*, indeed had the highest proportion of responses demonstrating this level.

The prompt "What are the consequences of an input validation vulnerability?" was developed to encourage *Making Meaning*, or putting together ideas. In Table VIII a response contains "inputs are accepted" and "break the program". 13.6% of responses to this prompt demonstrated *Making Meaning*. This is actually a smaller percentage than the other two prompts.

The prompt "What can you do to reduce the likelihood that you will write code with a input validation vulnerability?" was meant to encourage *Transformative Learning* or restructuring of ideas. The response in Table IX, "I will stop assuming people will input what I want...", indicates a restructuring of ideas. 30% of the responses to this prompt indicated *Transformative Learning*. This prompt produced the highest fraction of responses at this level. These results indicate several things. First, the classifier performs well, as demonstrated by a random check, and by the results that show that prompts meant to elicit responses at a particular level of learning follow the intended pattern. Second, "What is a possible consequence..." appears to a less effective prompt than "What is most surprising..." and "What can you do to reduce the likelihood...". "What is a possible consequence..." produced the lowest percentage of responses indicating *Making Meaning*, even though this prompt was meant to elicit this level of learning. The other prompts produce the highest percentages of the level of learning they were meant to elicit.

Over the three semesters, 1681 Computer Organization students completed the Buffer Overflow assignment and provided responses to reflection prompts. Tables XI-XIII show sample responses. Table XI shows the responses to the "What was most surprising" prompt. It is interesting to note that the example *Transformational Learning* response demonstrates the presence of new learning, so the classifier can find discussion of the future, and it can find discussion of discovery of new knowledge.

Table XIV shows the fraction of responses at each level

TABLE VII
CS1 "WHAT WAS MOST SURPRISING" SAMPLE RESPONSES

Level of Learning	**Example student response**
Making Sense (describing the problem)	Cross site scripting. I knew that *suspicious* links are not supposed to be clicked on, but I had no idea that it was such a *common* occurrence, nor did I know how *easily* it could be performed.
Making Meaning (integrating ideas)	I was surprised that something as *simple as checking* to make sure user input is fulfilling the assumptions you make about the input (such as height is positive), can *greatly improve the security* of your program.
Transformative Learning (restructuring of ideas)	I was suprised about just how common they are, this has pushed me to *start thinking about it when I begin to program future projects*.

TABLE VIII
CS1 "WHAT IS A POSSIBLE CONSEQUENCE?" SAMPLE RESPONSES

Level of Learning	**Example student response**
Making Sense	As shown in the Heartbleed attack and Whatsapp buffer flow, hackers are able to *infiltrate* private data and also *degrade* services across the internet.
Making Meaning	The possible consequence of input validation vulnerabilities is that some *inputs may be accepted when they are not supposed to be. It could break the program*, or even create an infinite loop. The point of input validation is to prevent the values that were never supposed to be inputted in the first place from going through the program.

TABLE IX
CS1 "WHAT CAN YOU DO TO REDUCE THE LIKELIHOOD" SAMPLE RESPONSES

Level of Learning	**Example student response**
Making Sense	Be *always aware* of the assumptions you make about the input you are accepting.
Making Meaning	Identify exactly what input you expect from a user, including input type and size. During the design process *find ways to counteract errors in the case that a user submits an invalid input*
Transformative Learning	*I will stop assuming* that people will input what I want them to input, such as for height assuming people would input a positive number and not NAN or infinity, and make sure it is the correct syntax, type, and meets the correct semantic criteria.

TABLE X
FRACTION OF RESPONSES AT EACH LEVEL OF LEARNING FOR CS1 INPUT VALIDATION ASSIGNMENT

Prompt	***Making Sense***	***Making Meaning***	***Transformative Learning***
What was most surprising	0.426	0.146	0.136
What is a possible consequence	0.045	0.136	0.0
What can you do to reduce the likelihood	0.272	0.195	0.300

of learning. 85.2% of responses to "What was the most surprising" prompt produced responses that demonstrated *Making Sense*. 88.2% of responses to "What can you do to reduce the likelihood of creating code vulnerable to buffer overflow" produced responses that demonstrated *Making Sense*. The students further along in the curriculum produce answers detailed with descriptive language after a hands-on assignment. The results indicate that the question meant to elicit *Making Meaning* produced a high percentage at that level, further giving confidence in the results of the classifier, and the appropriateness of the question.

88.8% of the responses to "What is a possible consequence of a buffer overflow vulnerability" produced responses that demonstrated *Making Meaning* in which multiple ideas were combined. This is the intended effect of this question, to combine ideas that develop the description of a consequence. 75.2% of responses to "What was the most surprising" prompt produced responses that *Making Meaning*. The Computer Organization students were overwhelmingly able to produce responses that put together ideas.

30.5% of responses to "What can you do to reduce the likelihood" produced responses that demonstrated *Transformative Learning*, and 53.3% of responses to "What was most surprising" demonstrated *Transformative Learning*. These responses indicate a change in ideas or plans for the future.

These results indicate that the prompts elicited responses at the level of reflection they were designed for. "What was most surprising..." elicited responses at all three levels of reflection. "What can you do to reduce the likelihood..." produced a large percentage of responses at *Making Sense* in addition to the intended *Transformative Learning*. The results also indicate that Computer Organization students overwhelmingly were able to produce responses indicating *Making Sense* and *Making Meaning*, and about half were able to produce responses indicating *Transformative Learning*.

VIII. DISCUSSION

To answer RQ1, the Cohen's Kappa values indicating substantial agreement between human coders and the NLP models and performance similar to previous results provides confidence that it is reasonable to use the NLP models to provide an estimate of the distribution of level of learning indicated in responses to reflection prompts. Reading through

TABLE XI
COMPUTER ORGANIZATION "WHAT WAS THE MOST SURPRISING" SAMPLE RESPONSES

Level of Learning	**Example student response**
Making Sense	The most interesting thing I learned is that even the *simplest* programs can be vulnerable to attacks and relying on code libraries to be safe can get you in trouble.
Making Meaning	The most surprising thing I learned was the fact that buffer overflow errors are very common and easy to have in programs. *One line of code in the program caused the buffer overflow to occur and it caused a cascade* of problems on the cache.
Transformative Learning	I found it interesting that if a buffer overflow vulnerability exists, one can access non-public code. We specifically used it to access an arbitrary function which was not called in main, *which I thought would never be possible*.

TABLE XII
COMPUTER ORGANIZATION "WHAT IS A CONSEQUENCE" SAMPLE RESPONSES

Level of Learning	**Example student response**
Making Sense	Buffer overflow can cause memory leaks and you can lose information. This loss of information can be *detrimental* to companies.
Making Meaning	The consequences of buffer overflow vulnerabilities can be severe. As we saw in the lab, when a 20 character input was used to overflow the 8 character buffer, it overwrote the address of the link return register x30. Someone with knowledge of the code they're manipulating can likely input a string of characters in such a way to do something malicious. For example in a banking system, *if they have a buffer overflow vulnerability in their login system, someone could potentially manipulate account data* doing nothing more than inputting strings.

TABLE XIII
COMPUTER ORGANIZATION "WHAT CAN YOU DO TO REDUCE THE LIKELIHOOD" SAMPLE RESPONSES

Level of Learning	**Example student response**
Making Sense	I can try to *be aware of "dangerous code"* that requires precision and foresight to prevent these types of mistakes. Also, I can make sure to test my code and be aware of edge cases.
Making Meaning	I would *try to avoid* commands that specifically are vulnerable to buffer overflow like fgets *now that I know* of the security risks involved. I would ensure that the size of the data written into a buffer is constrained to be within bounds of the buffer size so a buffer overflow cannot occur.
Transformative Learning	*I will test* for buffer overflow tests and also run them through tests that will see the memory registers and make sure that each register is being used correctly with the stack frame. I can also install software and do research on similar programs to prepare for probable reasons.

TABLE XIV
FRACTION OF RESPONSES AT EACH LEVEL OF LEARNING FOR COMPUTER ORGANIZATION BUFFER OVERFLOW ASSIGNMENT

Prompt	***Making Sense***	***Making Meaning***	***Transformative Learning***
What was most surprising	0.852	0.732	0.533
What is a possible consequence	0.166	0.888	0.0
What can you do to reduce the likelihood	0.881	0.218	0.305

the a random selection of the classification results indicates that the classifier can accurately reflect the ideas in Table I.

To answer RQ2, the results show that the prompts meant to elicit each stage of reflection did appear to work as intended according to the results from the NLP classifier. This indicates that NLP classification can be used to assess the effectiveness of reflection prompts. For the assignment given to CS1 students, the prompts for *Making Sense* and *Making Meaning* produced the highest percentage of responses at that level. 42% demonstrated *Making Sense* and 30% demonstrated *Transformative Learning*. The prompt for *Making Meaning* was less successful, and the other prompts actually produced more responses at that level. The highest percentage for *Making Meaning* was 19.5%. For the assignment given to Computer Organization students, the prompts for *Making Sense* and *Making Meaning* produced a high percentage of responses at the intended level of reflection: 85.2% and 88.8%, respectively. The prompt for *Transformative Learning* produced 39.5% of responses at that level, and the *Making Sense* prompt actually produced more *Transformative Learning* responses, 53.3%.

It is not surprising that *Transformative Learning* had lower percentages. The underlying theory of learning from Moon [4] states that stages of learning are reached consecutively. One cannot reach *Transformative Learning* without beforehand reaching *Making Meaning*. Thus any students indicating *Transformative Learning* must have reached *Making Meaning*. Barthakur et al. [15] performed a similar experiment in which machine learning was used to classify responses of professionals to guided reflection prompts in a workplace learning situation into the four stages of learning. That experiment found that about 90% of responses indicated *Making Sense*,

50% of responses indicated *Making Meaning* and 5% of responses indicated *Transformative Learning*. Different context and different prompts makes it difficult to compare, but the results suggest that the prompts used in this experiment produced high percentages at each level of learning, suggesting that the prompts are effective.

A much higher percentage of students in Computer Organization gave meaningful responses to reflection prompts, indicating movement along the stages of reflection toward *Transformative Learning*. There are several factors that contributed to this. These students are more mature; they are one year further along in the Computer Science curriculum. This is the second time they have been exposed to material on buffer overflow. The hands-on nature of the assignment also serves to elicit meaningful learning.

IX. Conclusion

This paper explored the use of NLP to classify responses to reflection prompts. A classifier for each of three levels of reflection, *Making Sense*, *Making Meaning*, and *Transformative Learning* showed substantial agreement with human coding. Spot checks of responses indicated that the classifiers correctly recognized descriptive words in *Making Sense*, multiple ideas in *Making Meaning*, and new learning in *Transformative Learning*. This suggests that NLP can be used to accurately classify levels of reflection.

Answering reflection prompts has been shown to help students meaningfully learn material. Assessing the responses can be time consuming for instructors. This work showed that NLP classifiers can be used to aid instructors in assessing the effectiveness of assignments and prompts in eliciting meaningful responses. The classification scheme described in this paper was developed in a Data Structures class and the NLP classifiers were trained with responses from a Data Structures class and a Computer Organization class. This suggests that the results may be generalizable across different CS courses.

The classification of responses indicates that the prompts elicited the levels of reflection they were intended to encourage. The experiment showed that more students in a second year course demonstrated the different levels of learning compared to students in a first year course. This suggests that NLP can use used to assess the effectiveness of reflection prompts and engagement of students.

References

[1] ACM, "Curriculum Guidelines for Undergraduate Degree Programs in Computer Science 2013."

[2] J. Dewey, *How we think: A restatement of the relation of reflective thinking to the educative process*. DC Health, 1933.

[3] D. A. Schon, *The reflective practitioner: How professionals think in action*. Basic books, 1984.

[4] J. Moon, "PDP Working Paper 4 Reflection in Higher Education Learning Jenny Moon , University of Exeter," *LTSN Generic Centre*, no. July, pp. 1–25, 2001, iSBN: 0202200485.

[5] R. Rogers R., "Reflection in Higher Education: A Concept Analysis," *Innovative Higher Education*, vol. 26, no. 1, pp. 37–57, 2001.

[6] M. G. Correia and R. E. Bleicher, "Making connections to teach reflection," *Michigan Journal of Community Service Learning*, vol. 14, no. 2, pp. 41–49, 2008, iSBN: ISSN-1076-0180.

[7] J. Coffey and B. Owsnicki, "Introducing a reflective activity into the design process in an advanced computer programming course," *Journal of Computing Sciences in Colleges*, vol. 31, no. 5, pp. 29–37, 2016.

[8] J. W. Coffey, "A Study of the Use of a Reflective Activity to Improve Students' Software Design Capabilities." Association for Computing Machinery (ACM), Mar. 2017, pp. 129–134.

[9] A. Nylén and V. Isomöttönen, "Exploring the critical incident technique to encourage reflection during project-based learning." Association for Computing Machinery (ACM), Nov. 2017, pp. 88–97.

[10] T. Caruso, N. Hill, T. Van DeGrift, and B. Simon, "Experience report: Getting novice programmers to THINK about improving their software development process," *SIGCSE'11 - Proceedings of the 42nd ACM Technical Symposium on Computer Science Education*, pp. 493–498, 2011, iSBN: 9781450305006.

[11] S. Kakavouli, P. T. Metaxas, P. Takis, and P. T. Metaxas, "Also Your Job to Learn! Helping Students to Reflect on their Learning Progress," *Journal of Computing Sciences in Colleges*, vol. 27, no. 6, 2012. [Online]. Available: http://repository.wellesley.edu/computersciencefacultyhttp://dl.acm.org/.

[12] M. Alzaid and I. H. Hsiao, "Effectiveness of Reflection on Programming Problem Solving Self-Assessments," *Proceedings - Frontiers in Education Conference, FIE*, vol. 2018-Octob, 2019, publisher: IEEE ISBN: 9781538611739.

[13] A. Pears and L. Larzon, "Student Perceptions of Reflections as an Aid to Learning," in *Proceedings of the 6th Baltic Sea Conference on Computing Education Research: Koli Calling 2006*. New York, NY, USA: Association for Computing Machinery, 2006, pp. 38–45, series Title: Baltic Sea '06. [Online]. Available: https://doi.org/10.1145/1315803.1315812

[14] C. Resch, P. Stapleton, B. Rheault, A. Wu, and C. Gardner-Mccune, "Analysis of effect of answering reflection prompts in a computer organization class," in *2022 ASEE Annual Conference*, 2022.

[15] A. Barthakur, S. Joksimovic, V. Kovanovic, R. F. Mello, M. Taylor, M. Richey, and A. Pardo, "Understanding depth of reflective writing in workplace learning assessments using machine learning classification," *IEEE Transactions on Learning Technologies*, vol. 15, no. 5, pp. 567–578, 2022.

[16] Y. R. Tausczik and J. W. Pennebaker, "The psychological meaning of words: Liwc and computerized text analysis methods," *J. Lang. Social Psychol.*, vol. 29, p. 24–54, 2010.

[17] D. McNamara, A. C. Graesser, P. M. McCarthy, and Z. Cai, *Automated Evaluation of Text and Discourse With Coh-Metrix*. Cambridge Univ. Press, 2014.

[18] J. Cohen, *Statistical power analysis for the behavioral sciences*. Academic press, 2013.

[19] P. Wulff, L. Mientus, A. Nowak, and A. Borowski, "Utilizing a pretrained language model (bert) to classify preservice physics teachers' written reflections," *International Journal of Artificial Intelligence in Education*, vol. 33, no. 3, pp. 439–466, 2023.

[20] C. Resch, A. Kapoor, and C. Gardner-Mccune, "Reflection and transformational learning in a data structures course," in *2021 ASEE Annual Conference*, 2021.

[21] A. V. Ganesan, M. Matero, Ravula, A. R., H. Vu, and H. A. Schwartz, "Empirical evaluation of pre-trained transformers for human-level nlp: the role of sample size and dimensionality.n," in *Proceedings of the Association for Computational Linguistics*. NIH Public Access, May 2021, p. 4515.

[22] "Hugging face: The ai community building the future," https://huggingface.co/, accessed: 2022-12-19.

[23] A. Gholamy, V. Kreinovich, and O. Kosheleva, "Why 70/30 or 80/20 relation between training and testing sets: A pedagogical explanation." 2018.

[24] C. Resch and C. Gardner-Mccune, "Giving students a view of buffer overflow with readily available tools," in *2023 ASEE Annual Conference*, 2023.