Autograding "Explain in Plain English" questions using NLP

Max Fowler
University of Illinois
Urbana, IL, USA
mfowler5@illinois.edu

Binglin Chen
University of Illinois
Urbana, IL, USA
chen386@illinois.edu

Sushmita Azad
University of Illinois
Urbana, IL, USA
sazad2@illinois.edu

Matthew West
University of Illinois
Urbana, IL, USA
mwest@illinois.edu

Craig Zilles
University of Illinois
Urbana, IL, USA
zilles@illinois.edu

ABSTRACT

Previous research suggests that "Explain in Plain English" (EiPE) code reading activities could play an important role in the development of novice programmers, but EiPE questions aren't heavily used in introductory programming courses because they (traditionally) required manual grading. We present what we believe to be the first automatic grader for EiPE questions and its deployment in a large-enrollment introductory programming course. Based on a set of questions deployed on a computer-based exam, we find that our implementation has an accuracy of 87–89%, which is similar in performance to course teaching assistants trained to perform this task and compares favorably to automatic short answer grading algorithms developed for other domains. In addition, we briefly characterize the kinds of answers that the current autograder fails to score correctly and the kinds of errors made by students.

CCS CONCEPTS

• **Social and professional topics → CS1**; • **Computing methodologies → Natural language processing**.

KEYWORDS

code reading, Explain in Plain English, ASAG, NLP, CS1

ACM Reference Format:
Max Fowler, Binglin Chen, Sushmita Azad, Matthew West, and Craig Zilles. 2021. Autograding "Explain in Plain English" questions using NLP. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (SIGCSE '21), March 13–20, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3408877.3432539

1 INTRODUCTION

The historic difficulty in learning to program [24] may be in part due to a pre-mature emphasis on code writing rather than code reading. For example, a recent theory of instruction for introductory programming [42] suggests that students should understand common programming patterns/idioms via code reading activities prior to being asked to compose those programming patterns/idioms to solve problems (i.e., writing complete routines). This theory is well motivated by the research literature (discussed in Section 2) which recognizes that (1) expert programmers use programming language features in idiomatic ways, (2) learning to program is as much about learning these idioms as it is about learning the syntactic elements themselves, and (3) activities like code tracing and code reading provide novices a means of developing programming skills with a lower cognitive load than code writing itself and, hence, should be employed to prepare students for code writing.

While code tracing and code writing activities are commonplace in introductory programming (CS 1) courses, code reading is not nearly as prominent. We could just ask our students to study large banks of programming examples, but many students require an incentive to get them to do assigned reading activities [14, 30]. One approach to gradable code reading activities proposed in the literature is "Explain in Plain English" (EiPE) questions (as shown in Figure 1) that ask students to describe in natural language (e.g., English) the high-level behavior of a short piece of code. While EiPE questions are well regarded by researchers, they aren't in widespread instructional use, presumably due to the burden of manually grading them and the slow feedback provided to students. In light of the large enrollments in many CS 1 classes, we sought to develop a means to automatically grade EiPE questions. As such, this paper poses the following research question: Can natural language processing (NLP) technology be used to automatically grade EiPE questions at accuracy rates comparable to humans?

In this work, we describe our initial development of an NLP-based autograder for EiPE questions and its use in a large-enrollment university-level introductory programming course. In particular we: 1) describe what we believe to be the first published implementation of autograded EiPE questions; 2) characterize the accuracy of the implementation as a function of the amount of training data, with our 87–89% accuracy suggesting EiPE may be simpler than other kinds of automatic short-answer grading; 3) compare the accuracy of the EiPE autograder with trained (non-researcher) teaching assistants (TAs) and find that the EiPE autograder has similar accuracy.

2 RELATED WORK

2.1 "Hierarchy" of programming skills

Researchers theorize that there is a loose hierarchy of programming skills with code writing at the top of the hierarchy, and many programming students struggle with tasks lower in the hierarchy [17].

SIGCSE '21, March 13–20, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-8062-1/21/03…$15.00
https://doi.org/10.1145/3408877.3432539

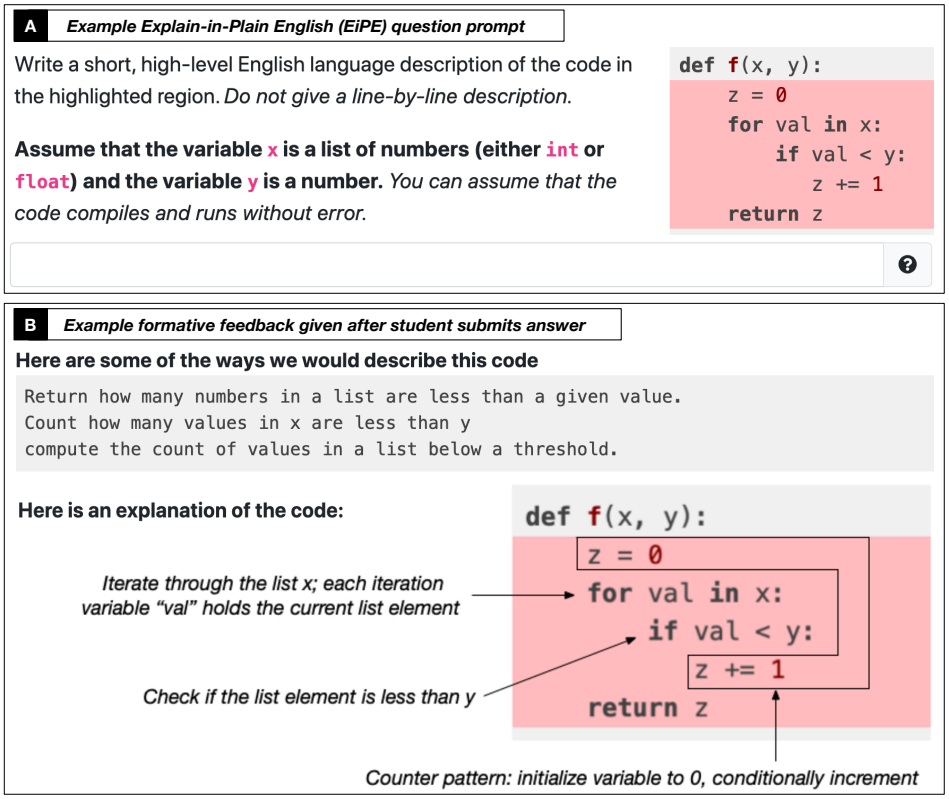

Figure 1: An example mid-semester automated code-reading exercise (A) in a Python-based intro CS course. After students submit their answers, they are graded and shown example solutions (B) to aid learning. Non-trivial code fragments are deconstructed—highlighting important idioms—to show correspondences between code and natural language descriptions.

These skills span from understanding syntax (as the easiest), to code tracing (executing code in your head for one particular input), to code reading/explaining (abstracting the behavior of code across all inputs), to code writing (as the most complex) [17]. It has been shown that for a given piece of code, task difficulty generally increases as we move up the hierarchy (e.g., tracing a swap vs. reading/explaining a swap vs. writing a swap) [20, 39], and students' mastery of the lower level skills is predictive of their code writing ability [8, 18, 20, 37]. In particular, Lopez et al. find that students' performance on tracing and code reading questions account for 46% of the variance in their performance on code writing questions [20]. Lister et al. state that, while their data doesn't support the idea of a strict hierarchy, "We found that students who cannot trace code usually cannot explain code, and also that students who tend to perform reasonably well at code writing tasks have also usually acquired the ability to both trace code and explain code."[18]

Whalley et al. argue that in order for a novice to write a particular piece of code, they must be able to comprehend that same piece of code and the knowledge and strategies within it [39]. In particular, programmers need to be able to understand code at the *relational* level (i.e., can summarize the code's purpose) and not just the *multistructural* (i.e., line-by-line) level [39]. Longitudinal studies find that students who are unable to explain code relationally early in the semester have difficulty writing code later in the semester [8].

It has been proposed that novice instruction should focus more on code tracing and reading [1, 6, 7, 18, 27, 40]. Lister et al. state, "It is our view that novices only begin to improve their code writing ability via extensive practice in code writing when their tracing and explaining skills are strong enough to support a systematic approach to code writing …" [18]

In particular, code reading activities may be among the best activities for helping novices learn to use common programming idioms. One important difference from novice programmers is that experts can automatically 'chunk' multiple syntax elements and process them as one unit [6, 12, 15, 25, 41], which reduces their cognitive load [35]. These chunks (or *schema* in the cognitive load literature) are developed through repeated experiences that have identifiable features in common [23] and are learned more efficiently in lower cognitive load activities (i.e., code reading rather than code writing) [36]. When students had relevant schema available, Rist found that students could and did reason forward from plan to code [31].

While all published works involving EiPE questions have been on summative assessments, Corney et al. suggest that "the value of EiPE problems may therefore be more as formative assessment rather than summative assessment" [7]. The challenges to using EiPE in formative assessment are having incentives in place to get the students to do the assignments [14] and giving the students immediate feedback on their understanding [29].

2.2 Automatic short answer grading (ASAG)

Automatically grading EiPE questions falls into the category of automatic short answer grading (ASAG). ASAG is characterized

Dataset	Score type	Best score	Citation
Texas [26]	r	0.630	[13]
ASAP-SAS [10]	QWK	0.791	[16]
Beetle UA 5-way [9]	micro F1	0.780	[22]
SciEntsBank UA 5-way [9]	F1	0.692	[22]

Table 1: The strongest reported results on publicly-available ASAG datasets [11].

by five criteria [3]: (1) student must recall external knowledge, (2) natural language responses, (3) response length of a phrase to a few sentences, (4) grading emphasizes content rather than writing style, and (5) focused prompts. ASAG is challenging because there are many ways to express a correct answers in natural language.

While there has been an evolution of ASAG techniques [3], current competitive ASAG systems are all based on machine learning [11]. These use a variety of features, including lexical (e.g., bag-of-words), morphological (e.g., stem matches), semantic (e.g., latent semantic analysis), syntactic (e.g., part-of-speech tagging), and surface (e.g., word count). A wide variety of machine learning techniques have also been used [11]. Most recently, dialog based systems and intelligent tutoring systems [28, 32, 33] and end-to-end models have been used for ASAG [19, 43].

Much of the work on ASAG is focused on K-12, so there is relatively little ASAG work on Computer Science subject matter. The main application of ASAG on computing content is the "Texas dataset" [26], which features traditional short answer questions from a data structures course (e.g. "What is the role of a prototype program in problem solving?") and no questions related to describing code. Many researchers have used this data set to evaluate novel ASAG algorithms [13, 21, 34], but none of these papers are computer science domain specific.

The current state of the art of ASAG is only modestly accurate. Four publicly-available ASAG datasets have multiple results reported [9, 10, 26] (see [11] for a detailed summary). Table 1 lists the best reported results for ASAG datasets (as of early 2020).

3 AUTOGRADED EIPE QUESTIONS

In line with research literature recommendations (Section 2.1), we developed our EiPE system primarily for use as a formative assessment (e.g., homework), as our primary goal was repeated practice with reading code so that students would incorporate common programming idioms into their programming repertoire. As will be discussed in Section 4, we do include the EiPE questions on our exams, primarily to motivate students to take this formative activity seriously and less to evaluate students skill level.

Our EiPE activities have been implemented in the same online system that the students use for their homework. An example EiPE question is shown in Figure 1(A). In general, the questions have been designed to have relational descriptions, but some of the questions in the first couple of weeks of the semester involve just a single operator (e.g., concatenation, floor division) in order to introduce students to the tool early and ensure they know the names of operators. The components of this interface (e.g., directions, highlighting, type information) have been refined iteratively to direct students toward the desired kind of (relational) answer.

Student submissions are immediately scored, and student feedback includes example descriptions, as shown in Figure 1(B). For the later, more complicated questions, where we are concerned that some students might not be able to understand the relationship between the correct description with the provided code, the solution provides a visual explanation of how the code relates to the description, highlighting common programming idioms.

When placed on a weekly homework assignment, EiPE questions were grouped into pools of 8–12 questions. Each time a student attempts the question, they are shown a random selection from the question pool. To get full credit on the question, students have to answer a number of questions correctly. When they correctly answer the question, they are awarded points, but there is no penalty for incorrect answers. This approach makes the activity somewhat tolerant of an imperfect autograder, as any false negatives only delay (rather than prevent) students from getting their points.

Because there was no penalty for wrong answers, we discovered that a group of students were putting in "garbage" answers to get the system to show its example correct answers. We attempted to discourage this behavior in two ways: 1) we created a *garbage filter* that prevents the display of our correct answers when the student submission is not deemed to be an actual attempt to answer the question, and 2) we turned off text selection for the correct answers shown to make copying them require additional effort.

3.1 The EiPE autograder

The autograder that we report here is significantly simpler than the state of the art ASAG. Our autograder is a logistic regression model on bag-of-words and bigram (pairs of adjacent words) features. In order to minimize the impact of non-word symbols on feature extraction, students' responses need to be preprocessed.

The first step of preprocessing is to remove quotes in the students' responses. This helps to handle cases where students quoted a variable name or the entire response. The second step is to insert spaces before and after mathematical operators such as the addition operator "+" and all kinds of brackets. This ensures that all mathematical expressions will be spaced equivalently between operators and operands.

Each response is then split into sentences using the Python natural language toolkit `nltk`'s sentence tokenizer. Each sentence is further split into a list of tokens by `nltk`'s word tokenizer. The resulting tokens could be words, numbers, or math symbols. These tokens are then cast to lower case and passed to a spellchecker to correct misspelled words. At the end of the above processing, each student's response has been converted into lists of tokens.

During training, we select a set of bag-of-words and bigram features for each question individually. For each response in the training set, we first identify the set of words and bigrams present (ignoring duplicates). We then count what fraction of the training set has each word and bigram. The 40% most frequent words and 20% most frequent bigrams are then used as the feature set.

Each response is converted to a feature vector of 0s and 1s, where 1 at a index indicates a particular word or bigram in the feature set is present in the response. These vectors are then fed into `scikit-learn`'s logistic regression to train a model. The model produces binary correct/incorrect judgments.

The autograder's garbage filter is a rule-based algorithm relying on a vocabulary extracted from students' responses. Its vocabulary is created by tokenizing all correct student responses and counting

the occurrences of each token. The 75% most frequent tokens were considered as the *non-garbage* vocabulary. The garbage filter categorizes as garbage any response with two or fewer tokens or with less than 60% of its tokens in the non-garbage vocabulary.

4 METHODS

Our EiPE autograder was developed for and deployed in an introductory CS course for non-technical majors at a large U.S. university. This large-enrollment course (capped at 600 students) introduces basic principles of programming in both Python and Excel to a population largely without any prior programming experience. In this paper, we report results from both the Fall 2019 and Spring 2020 semesters. The course is taken predominantly by freshmen (67%) and sophomores (21%) and approaches gender balance (Fall 2019: 246 women/355 men; Spring 2020 258 women/307 men).

The autograding EiPE questions were deployed both on homework assignments during the first half of the semester and on proctored computer-based exams in each semester. We present results drawn only from student responses on exams because we can be assured that those results represent serious attempts by the students; we observed students attempting to "game the system" on the low-stakes homework, as discussed in Section 3.1. In Fall 2019, the EiPE questions appeared on a mid-term exam in the 12th week and, in Spring 2020, the EiPE questions appeared on a mid-term exam in the 6th week of the course. This discrepancy is due to the technology still being refined during the Fall 2019 semester.

In both cases, we used the pool of EiPE questions deployed on the homework during the 5th week of the course. Students took their exams in a Computer-Based Testing Facility [44] over a three-day period. To minimize the potential for cheating on these asynchronous exams, we randomly assigned students questions from the question pool on an individual basis [5]. Four of the problems in the pool were not included on the exam because they were significantly easier or harder than the rest.

In Fall 2019, students were given three attempts to submit a correct answer (the exam is graded interactively). Because we found that these multiple attempts resulted in additional false positives [2], in Spring 2020 we gave students a single attempt to answer the question. The students in Fall 2019 submitted a total of 1,140 responses and the students in Spring 2020 submitted a total of 582 responses.

4.1 Training Data and Ground Truth

The autograders were initially trained with data from a series of surveys. Each survey asked participants to provide two correct responses and two plausible incorrect responses for each EiPE question. These surveys were completed by the course's instructor, TAs, and a collection of upper-level CS students. Students were compensated with Amazon gift cards. These surveys resulted in approximately 100–200 responses per question. Survey data was manually reviewed by a research team member to perform any necessary re-categorization of the responses. Our binary scores corresponded to 0-4 (incorrect) and 5-6 (correct) on a previously validated scoring rubric [4].

Prior to deployment on the exam, two members of the research team manually labeled the Fall 2019 students' homework responses to these questions and used that as additional training data to improve the grader. The AI graders deployed on the exam were trained with 500–600 labeled responses per question.

Once an exam was completed, two members of the research team familiar with the course content scored each student response as correct/incorrect independent of the AI grader's score. Correct responses must be correct at the relational level (non line-by-line descriptions) and unambiguous. Responses were not penalized for spelling or grammar as long as the intent was clear. If the two scorers agreed on a response, this was considered the final ground truth. For any disagreements between the scorers, ground truth was established by a process of discussion and reconciliation between both scorers and a third research team member until consensus was reached. The inter-rater reliability of the two research team members' initial scores was a Cohen's kappa of 0.83 ("almost perfect" agreement [38]). Grades were manually adjusted for any students who were incorrectly denied credit by the autograder.

4.2 Training human graders

Because members of the research team have scored thousands of EiPE responses and spent tens of hours reconciling challenging responses and developing a shared understanding of how to handle corner-cases, our level of experience and interest in EiPE grading far exceeds that of the teaching assistants (TAs) who would perform manual grading in the context of a large-enrollment course. As such, we perform our comparison to manual grading using an ecologically valid population: the TAs for the Spring 2020 offering of the course in which the data was collected. The eight TAs that participated were all graduate students in computer science.

To investigate differences in experience, the eight TAs were split into two groups of four ("Trained TAs" and "Minimally-trained TAs") for different training regimens. The TAs were unaware of this split. IRB approval and TA consent were obtained for using their grading results.

Trained TAs: The four TAs in the "Trained TAs" group were given a joint 30-minute training session by a member of the research group. The training consisted of explaining the scoring criteria (e.g., high-level, correct, and unambiguous) to them in detail and then having them score example responses drawn from the Fall 2019 data in multiple rounds. Each round, the TAs independently scored five responses. Discrepancies between any of the TAs scores and the research team's ground truth scores were collectively discussed and resolved before continuing on to the next round. There were four such rounds; each round used responses from a different question per round. We assigned the three returning Fall 2019 TAs to Group A because they had some prior exposure to the EiPE questions during that offering of the course.

Minimally-trained TAs: The four TAs in the "Minimally-trained TAs" group were all new to the course. They were given only a 5-minute training session by the same member of the research group, during which they were only shown the scoring criteria. No examples were worked through or discussed.

TA grading: Following the training, all eight TAs scored a selection of EiPE student responses from the Spring 2020 exam, without access to either the ground truth or the model's score. Each TA received the same 100 responses for manual grading. The responses were stratified to disproportionately sample responses that the

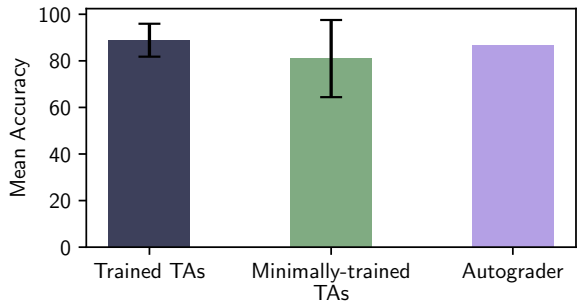

Figure 2: The mean accuracies with 95% confidence intervals specified alongside the Spring 2020 model accuracy.

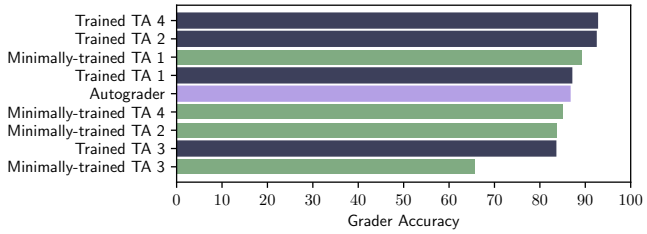

Figure 3: The accuracy of the autograder is in the same ballpark as the TAs (Spring 2020 data).

model struggled with; both False Negatives (FNs) and False Positives (FPs). All the available FNs were sampled, while FPs were sampled at about twice the rate of the True Postives (TPs) and True Negatives (TNs). The TA responses were reweighed from the sample back to the full population rates for each category. These results were compared to the model's performance on the Spring 2020 exam.

5 RESULTS

On the Fall 2019 exam, 67% of the students answered their EiPE question correctly and the autograder achieved an accuracy of 89%, with a 12% False Positive (FP) rate and a 9% False Negative (FN) rate. In Spring 2020, 45% of the students answered their EiPE question correctly and the autograder achieved an accuracy of 87%, with a 15% FP rate and a 10% FN rate. The model used a different threshold for marking answers correct in Fall 2019 and Spring 2020. We found that running Fall 2019's responses using Spring 2020's threshold reduced accuracy from 89% to 88%.

Mean accuracies for the model, trained TAs, and minimally-trained TAs are plotted in Figure 2. We did not find a statistically significant difference between the model and TAs, regardless of training. The model did perform about as well as three of our trained TAs and outperformed all but one minimally trained TA. These results, however, are not statistically significant because of the small number of TAs ($N = 4$ in each case). The TAs' individual accuracies are sorted and plotted in Figure 3. Inter-rater reliability, F1 score, and accuracy relative to the ground truth for each TA individually are presented in Table 2.

Table 3 shows these results in the same metrics reported by the best current results on publicly-available ASAG data sets shown in Table 1. While clearly not directly comparable due to different data sets, we note that our model is achieving higher accuracy in spite of the fact that it is much less sophisticated. As we discuss in Section 6, we believe that this indicates that EiPE autograding is a simpler task than ASAG in general.

Grader	Accuracy	κ	F1
Trained TA 1	93%	0.834	0.921
Trained TA 2	92%	0.831	0.920
Trained TA 3	87%	0.708	0.845
Trained TA 4	83%	0.626	0.766
Minimally-trained TA 1	89%	0.776	0.889
Minimally-trained TA 2	84%	0.681	0.833
Minimally-trained TA 3	85%	0.698	0.832
Minimally-trained TA 4	66%	0.383	0.689
Autograder model	87%	0.718	0.826

Table 2: Comparison of the trained TA, minimally-trained TA, and Spring 2020 model performance.

Score type	FA19 (12th week)	SP20 (6th week)
r	**0.748**	**0.725**
QWK	0.740	0.718
accuracy (micro F1)	**0.888**	**0.866**
F1	**0.820**	**0.826**

Table 3: The autograder's results for Fall 2019 and Spring using the metrics in Table 1. Bolded scores are higher than corresponding metrics in Table 1.

To understand how much training data is needed to obtain a reasonable AI grader and whether there is a qualitative difference between survey data and student homework data, we trained graders with different subsamples of data and show the mean of the grader's performance on the Fall 2019 responses in Figure 4. There are three sources of training data: (1) a subset of the survey data, (2) a subset of the student homework data, and (3) both, meaning all of the survey data and a subset of the student homework data. Although more data consistently led to better performance, the student homework data seems qualitatively better than survey data, suggesting that the course staff and senior students creating the survey data were only somewhat able to generate realistic training data.

5.1 Characterizing Model Failure

Discussion of False Negatives: The model primarily struggles with two types of questions. First, in binary conditionals (e.g., "Is a number even?"), the model struggles with responses like "whether the variable is even or odd". This response has both a 'correct' keyword (even) and an 'incorrect' keyword (odd), which presents conflicting signals to the autograder. Human graders however, relying on colloquial English usage, would interpret such responses as '(*Returns*) whether the variable is even (*True*) or odd (*False*)', and score them as correct.

Second, in bi-variate comparisons ("Max of two numbers", "Is x a factor of y?") the autograder seems to struggle with recognition of uncommonly used synonyms and syntax. For example, the answer "returns the largest variable" is incorrectly marked wrong, but for all such responses, when we change the word 'variable' to 'number', then the model scores them as correct. These miscategorizations seem to be due to few correct responses in the training data that use the word 'variable'. In addition, the model struggles with unnatural

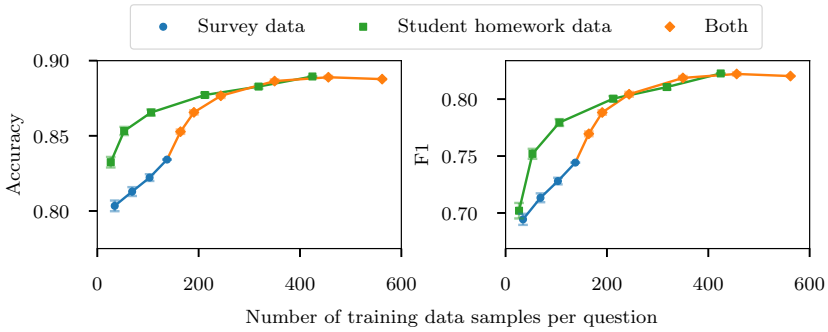

Figure 4: The performance of the AI grader on the 1,140 exam responses from Fall 2019, when trained on different combinations of data with different sample sizes. Error bars are 95% confidence intervals.

phrasing; for example, in "return its smallest number" it is unclear what "it" refers to. Minor typos, however, do not result in FNs.

Discussion of False Positives: We see both a larger number of FPs (101 FPs vs 29 FNs) as well as more diversity in the types of questions and responses that caused them. In general, as long as the student response has the *keyword* or a set of crucial words that appear in the training set of correct responses and does not have the corresponding *incorrect keyword* (e.g. 'odd' when looking for 'even'), the model tends to score it as correct. Extra information or word order leading to an incorrect meaning tend to be ignored. This is likely due to our model using a bigram (+ bag-of-words) approach. Bigrams are not able to capture negative signals when negative signals are more than two words long and the bigrams in that negative signal have also appeared in a correct answer. For this high stakes exam deployment, we biased the model to avoid false negatives [2]. While the FP rate at 12% is higher than we would like, it should decrease with model improvement.

6 DISCUSSION

We were genuinely surprised that our very simple model performed as well as it did. We initially intended the bag-of-words + bigrams model to be a temporary model that we'd only use long enough in order to collect additional training data, so that we could train a more sophisticated model. Instead, we found that the model was good enough to allow us to focus on other parts of introducing EiPE questions in the course, including authoring additional questions and implementing the garbage filter. Furthermore, we deemed it accurate enough that we were willing to use it for interactive grading on an exam, with the knowledge that we would manually re-grade the student answers after the exam [2].

From these results it appears that automatically grading EiPE questions may be a simpler task than other ASAG contexts. Even using just bigrams, our results compare favorably with other ASAG results using much more sophisticated algorithms. We believe that this high accuracy is the result of specific elements of disciplinary vocabulary (e.g., "count", "even") being effective markers of when students have correct answers. We are optimistic that accuracy can be further improved through the introduction of state of the art ASAG techniques. Many of the errors that the model makes can be attributed to its limited ability to exploit word ordering. A more sophisticated model will likely require additional training data.

The data in Figure 4 supports the assertion that the current model is not limited by the amount of training data. The model's accuracy is almost constant once we reached 350 items in the training set. Furthermore, it is not surprising that the student homework responses were more effective than survey data for training the algorithm to predict student exam responses. The surveys did enable us to deploy the algorithm in the low stakes homework context to collect that homework training data, but our conclusion is that we could get by with fewer survey responses, especially if we were to quickly score early homework responses and re-train the model.

Our current implementation's accuracy is in the same ballpark as our average course TAs, even after training. We believe that even just using the state of the art ASAG techniques should enable an algorithm to be competitive with even our most accurate TAs.

7 CONCLUSION

We believe this paper reports on the first instance of autograding "Explain in Plain English" (EiPE) questions. It appears that autograding EiPE questions is easier than the more general problem of automatic short answer grading. By training a simple model (bag-of-words + bigrams) using student homework responses to EiPE questions, we were able to achieve similar accuracy in autograding EiPE questions to manual grading by course TAs.

We believe that autograded EiPE questions have the potential to significantly impact instruction in introductory programming courses. Many researchers have advocated for a greater emphasis on code reading activities, and automated EiPE activities can effectively scale to even large enrollment CS 1 courses.

There are many directions for future work. First, recent advances in ASAG and NLP more broadly should be introduced into the model. Second, we suggest evaluating EiPE autograding on a broader range of questions to better characterize how autograding accuracy varies across questions and populations. Such an analysis might provide insight into how to further improve the algorithm. Finally, once a mature autograding EiPE system is available, it will be important to identify best practices for adopting it into existing courses and measuring the impact a larger emphasis on code reading has on introductory programming instruction.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under grant numbers DUE-1347722 and DUE-1915257.

REFERENCES

[1] Owen Astrachan and David Reed. 1995. AAA and CS 1: The Applied Apprenticeship Approach to CS 1. In *Proceedings of the Twenty-sixth SIGCSE Technical Symposium on Computer Science Education (SIGCSE '95)*. ACM, New York, NY, USA, 1–5. https://doi.org/10.1145/199688.199694

[2] Sushmita Azad, Binglin Chen, Maxwell Fowler, Matthew West, and Craig Zilles. 2020. Strategies for Deploying Unreliable AI Graders in High-Transparency High-Stakes Exams. In *International Conference on Artificial Intelligence in Education*. Springer, 16–28.

[3] Steven Burrows, Iryna Gurevych, and Benno Stein. 2015. The Eras and Trends of Automatic Short Answer Grading. *International Journal of Artificial Intelligence in Education* 25, 1 (01 Mar 2015), 60–117. https://doi.org/10.1007/s40593-014-0026-8

[4] Binglin Chen, Sushmita Azad, Rajarshi Haldar, Matthew West, and Craig Zilles. 2020. A Validated Scoring Rubric for Explain-in-Plain-English Questions. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE)*.

[5] Binglin Chen, Matthew West, and Craig Zilles. 2018. How Much Randomization is Needed to Deter Collaborative Cheating on Asynchronous Exams?. In *Learning at Scale*.

[6] Michael J. Clancy and Marcia C. Linn. 1999. Patterns and Pedagogy. In *The Proceedings of the Thirtieth SIGCSE Technical Symposium on Computer Science Education (SIGCSE '99)*. ACM, New York, NY, USA, 37–42.

[7] Malcolm Corney, Sue Fitzgerald, Brian Hanks, Raymond Lister, Renee McCauley, and Laurie Murphy. 2014. 'Explain in Plain English' Questions Revisited: Data Structures Problems. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education (SIGCSE '14)*. ACM, New York, NY, USA, 591–596. http://doi.acm.org/10.1145/2538862.2538911

[8] Malcolm Corney, Raymond Lister, and Donna Teague. 2011. Early Relational Reasoning and the Novice Programmer: Swapping As the "Hello World" of Relational Reasoning. In *Proceedings of the Thirteenth Australasian Computing Education Conference - Volume 114 (ACE '11)*. 95–104.

[9] M. O. Dzikovska et al. 2013. SemEval-2013 task 7: The joint student response analysis and eighth recognizing textual entailment challenge. In *Proceedings of the 2nd joint conference on lexical and computational semantics*, M. Diab, T. Baldwin, and M. Baroni (Eds.). 1–12.

[10] Hewlett Foundation. 2012. Automated student assessment prize: Phase two — short answer scoring, Kaggle Competition.

[11] Lucas Busatta Galhardi and Jacques Duílio Brancher. 2018. Machine Learning Approach for Automatic Short Answer Grading: A Systematic Review. In *Advances in Artificial Intelligence - IBERAMIA 2018*, Guillermo R. Simari, Eduardo Fermé, Flabio Gutiérrez Segura, and José Antonio Rodríguez Melquiades (Eds.). Springer International Publishing, Cham, 380–391.

[12] Fernand Gobet, Peter CR Lane, Steve Croker, Peter CH Cheng, Gary Jones, Iain Oliver, and Julian M Pine. 2001. Chunking mechanisms in human learning. *Trends in cognitive sciences* 5, 6 (2001), 236–243.

[13] Wael Hassan Gomaa and Aly Aly Fahmy. 2020. Ans2vec: A Scoring System for Short Answers. In *The International Conference on Advanced Machine Learning Technologies and Applications (AMLTA2019)*, Aboul Ella Hassanien, Ahmad Taher Azar, Tarek Gaber, Roheet Bhatnagar, and Mohamed F. Tolba (Eds.). Springer International Publishing, Cham, 586–595.

[14] Sarah J. Hatteberg and Kody Steffy. 2013. Increasing Reading Compliance of Undergraduates: An Evaluation of Compliance Methods. *Teaching Sociology* 41, 4 (2013), 346–352. https://doi.org/10.1177/0092055X13490752

[15] Vighnesh Iyer and Craig Zilles. 2021. Pattern Census: A Characterization of Pattern Usage in Early Programming Courses. In *Proceedings of the SIGCSE Technical Symposium (SIGCSE)*.

[16] Yaman Kumar, Swati Aggarwal, Debanjan Mahata, Rajiv Ratn Shah, Ponnurangam Kumaraguru, and Roger Zimmermann. 2019. Get IT Scored Using AutoSAS — An Automated System for Scoring Short Answers. *Proceedings of the AAAI Conference on Artificial Intelligence* 33, 01 (July 2019), 9662–9669. https://doi.org/10.1609/aaai.v33i01.33019662

[17] Raymond Lister, Elizabeth S Adams, Sue Fitzgerald, William Fone, John Hamer, Morten Lindholm, Robert McCartney, Jan Erik Moström, Kate Sanders, Otto Seppälä, Beth Simon, and Lynda Thomas. 2004. A multi-national study of reading and tracing skills in novice programmers. *ACM SIGCSE Bulletin* 36, 4 (2004), 119–150.

[18] Raymond Lister, Colin Fidge, and Donna Teague. 2009. Further Evidence of a Relationship Between Explaining, Tracing and Writing Skills in Introductory Programming. In *Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE '09)*. ACM, New York, NY, USA, 161–165. https://doi.org/10.1145/1562877.1562930

[19] Tiaoqiao Liu, Wenbiao Ding, Zhiwei Wang, Jiliang Tang, Gale Yan Huang, and Zitao Liu. 2019. Automatic Short Answer Grading via Multiway Attention Networks. *arXiv:1909.10166 [cs]* (2019). http://arxiv.org/abs/1909.10166

[20] Mike Lopez, Jacqueline Whalley, Phil Robbins, and Raymond Lister. 2008. Relationships between reading, tracing and writing skills in introductory programming. In *Proceedings of the Fourth International Workshop on Computing Education Research*. ACM, 101–112.

[21] Ahmed Magooda, Mohamed A. Zahran, Mohsen Rashwan, Hazem M. Raafat, and Magda B. Fayek. 2016. Vector Based Techniques for Short Answer Grading. In *FLAIRS Conference*.

[22] Alvarado Mantecon and Jesus Gerardo. 2019. *Towards the Automatic Classification of Student Answers to Open-ended Questions*. Thesis. Université d'Ottawa / University of Ottawa. https://doi.org/10.20381/ruor-23341

[23] Sandra P Marshall. 1995. *Schemas in problem solving*. Cambridge University Press.

[24] Michael McCracken et al. 2001. A Multi-national, Multi-institutional Study of Assessment of Programming Skills of First-year CS Students. In *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education (ITiCSE-WGR '01)*. ACM, New York, NY, USA, 125–180.

[25] Katherine B McKeithen, Judith Spencer Reitman, Henry H Rueter, and Stephen C Hirtle. 1981. Knowledge organization and skill differences in computer programmers. *Cognitive Psychology* 13, 3 (1981), 307–325.

[26] M. Mohler, R. Bunescu, and R. Mihalcea. 2011. Learning to grade short answer questions using semantic similarity measures and dependency graph alignments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. 752–762.

[27] Laurie Murphy, Renée McCauley, and Sue Fitzgerald. 2012. 'Explain in Plain English' Questions: Implications for Teaching. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (SIGCSE '12)*. ACM, New York, NY, USA, 385–390. https://doi.org/10.1145/2157136.2157249

[28] Ifeanyi G. Ndukwe, Ben K. Daniel, and Chukwudi E. Amadi. 2019. A Machine Learning Grading System Using Chatbots. In *Artificial Intelligence in Education (Lecture Notes in Computer Science)*. Springer International Publishing, Cham, 365–368.

[29] Bertram Opitz, Nicola K Ferdinand, and Axel Mecklinger. 2011. Timing matters: the impact of immediate and delayed feedback on artificial language learning. *Frontiers in human neuroscience* 5 (2011), 8.

[30] Alexander Renkl, Robin Stark, Hans Gruber, and Heinz Mandl. 1998. Learning from Worked-Out Examples: The Effects of Example Variability and Elicited Self-Explanations. *Contemporary educational psychology* 23 (01 1998), 90–108. https://doi.org/10.1006/ceps.1997.0959

[31] Robert S Rist. 1989. Schema creation in programming. *Cognitive Science* 13, 3 (1989), 389–414.

[32] Swarnadeep Saha, Tejas I. Dhamecha, Smit Marvaniya, Renuka Sindhgatta, and Bikram Sengupta. 2018. Sentence Level or Token Level Features for Automatic Short Answer Grading?: Use Both. In *Artificial Intelligence in Education (Lecture Notes in Computer Science)*. Springer International Publishing, Cham, 503–517.

[33] Chul Sung, Tejas Indulal Dhamecha, and Nirmal Mukhi. 2019. Improving Short Answer Grading Using Transformer-Based Pre-training. In *Artificial Intelligence in Education*. Vol. 11625. Springer International Publishing, Cham, 469–481.

[34] Neslihan Suzen, Alexander Gorban, Jeremy Levesley, and Evgeny Mirkes. 2019. Automatic Short Answer Grading and Feedback Using Text Mining Methods. *CoRR* (2019). http://arxiv.org/abs/1807.10543 arXiv: 1807.10543.

[35] John Sweller. 2011. Cognitive Load Theory. In *Psychology of learning and motivation*. Vol. 55. Elsevier, 37–76.

[36] John Sweller, Jeroen JG Van Merrienboer, and Fred GWC Paas. 1998. Cognitive architecture and instructional design. *Educational psychology review* 10, 3 (1998), 251–296.

[37] Anne Venables, Grace Tan, and Raymond Lister. 2009. A Closer Look at Tracing, Explaining and Code Writing Skills in the Novice Programmer. In *Proceedings of the Fifth International workshop on Computing Education Research*. ACM, 117–128.

[38] Anthony J Viera, Joanne M Garrett, et al. 2005. Understanding interobserver agreement: the kappa statistic. *Fam med* 37, 5 (2005), 360–363.

[39] Jacqueline Whalley, Raymond Lister, Errol Thompson, Tony Clear, Phil Robbins, P K Ajith Kumar, and Christine Prasad. 2006. An Australasian study of Reading and Comprehension Skills in Novice Programmers, using the Bloom and SOLO Taxonomies. *Eighth Australasian Computing Education Conference (ACE2006)* (2006).

[40] Susan Wiedenbeck. 1985. Novice/expert differences in programming skills. *International Journal of Man-Machine Studies* 23, 4 (1985), 383 – 390. https://doi.org/10.1016/S0020-7373(85)80041-9

[41] Leon E. Winslow. 1996. Programming Pedagogy— a Psychological Overview. *SIGCSE Bull.* 28, 3 (Sept. 1996), 17–22. https://doi.org/10.1145/234867.234872

[42] Benjamin Xie, Dastyni Loksa, Greg L Nelson, Matthew J Davidson, Dongsheng Dong, Harrison Kwik, Alex Hui Tan, Leanne Hwa, Min Li, and Andrew J Ko. 2019. A theory of instruction for introductory programming skills. *Computer Science Education* 29, 2-3 (2019), 205–253.

[43] Xi Yang, Yuwei Huang, Fuzhen Zhuang, Lishan Zhang, and Shengquan Yu. 2018. Automatic Chinese Short Answer Grading with Deep Autoencoder. In *Artificial Intelligence in Education (Lecture Notes in Computer Science)*. Springer International Publishing, Cham, 399–404.

[44] Craig Zilles, Matthew West, Geoffrey Herman, and Timothy Bretl. 2019. Every university should have a computer-based testing facility. In *Proceedings of the 11th International Conference on Computer Supported Education (CSEDU)*.