Computers and Education: Artificial Intelligence 3 (2022) 100081

Contents lists available at ScienceDirect

Computers and Education: Artificial Intelligence

journal homepage: www.sciencedirect.com/journal/computers-and-education-artificial-intelligence

Machine learning based feedback on textual student answers in large courses

Jan Philip Bernius *, Stephan Krusche, Bernd Bruegge

Department of Informatics, Technical University of Munich, Boltzmannstraße 3, 85748, Garching Near Munich, Germany

ARTICLE INFO

Keywords:
Software engineering
Education
Interactive learning
Automatic assessment
Grading
Assessment support system
Learning
Feedback

ABSTRACT

Many engineering disciplines require problem-solving skills, which cannot be learned by memorization alone. Open-ended textual exercises allow students to acquire these skills. Students can learn from their mistakes when instructors provide individual feedback. However, grading these exercises is often a manual, repetitive, and time-consuming activity. The number of computer science students graduating per year has steadily increased over the last decade. This rise has led to large courses that cause a heavy workload for instructors, especially if they provide individual feedback to students. This article presents CoFee, a framework to generate and suggest computer-aided feedback for textual exercises based on machine learning. CoFee utilizes a segment-based grading concept, which links feedback to text segments. CoFee automates grading based on topic modeling and an assessment knowledge repository acquired during previous assessments. A language model builds an intermediate representation of the text segments. Hierarchical clustering identifies groups of similar text segments to reduce the grading overhead. We first demonstrated the CoFee framework in a small laboratory experiment in 2019, which showed that the grading overhead could be reduced by 85%. This experiment confirmed the feasibility of automating the grading process for problem-solving exercises. We then evaluated CoFee in a large course at the Technical University of Munich from 2019 to 2021, with up to 2, 200 enrolled students per course. We collected data from 34 exercises offered in each of these courses. On average, CoFee suggested feedback for 45% of the submissions. 92% (Positive Predictive Value) of these suggestions were precise and, therefore, accepted by the instructors.

1. Introduction

Student numbers in computer science schools and departments are rising. Analyzing statistics and reports released by popular computer science departments reveals how the number of conferred degrees has steadily increased since 2010. Fig. 1 depicts the development of degrees conferred by eight renowned universities[1] in the area of computer science. As a result, introductory courses need to handle more and more students every year. This rise in student numbers has increased course management efforts and made it challenging to provide high-quality individual feedback to students (Krusche et al., 2020). A single instructor cannot handle feedback and grading for large classes alone. In particular, large university courses with hundreds of students rely on teaching assistants to provide feedback on exercises. Online platforms, live streaming, and chat systems allow instructors to interact with a large number of students on an individual level, regardless of the respective course size.

Exercises allow students in lecture-based courses to apply and practice relevant skills. Exercises stimulate learning in six different cognitive processes, e.g., as classified in Bloom's revised taxonomy (Anderson et al., 2001). Software engineering is a problem-solving discipline that cannot be learned by memorization alone.

Abbreviations: CoFee, Computer-aided Feedback for textual exercises; ELMo, Embeddings from Language Models; HDBSCAN, Hierarchical Density-Based Spatial Clustering of Applications with Noise; ISE, Introduction to Software Engineering; LSA, Latent Semantic Analysis; NLP, Natural Language Processing; POM, Project Organization and Management; PPV, Positive Predictive Value; TF-IDF, Term Frequency-Inverse Document Frequency; TNR, True Negative Rate; TPR, True Positive Rate; TUM, Technical University of Munich.

* Corresponding author.
E-mail addresses: janphilip.bernius@tum.de (J.P. Bernius), krusche@in.tum.de (S. Krusche), bernd.bruegge@tum.de (B. Bruegge).

[1] The universities were selected based on the *Times Higher Education Ranking* by Subject in 2022 and the availability of data. https://timeshighereducation.com/world-university-rankings/2022/subject-ranking/computer-science.

https://doi.org/10.1016/j.caeai.2022.100081
Received 6 March 2022; Received in revised form 24 May 2022; Accepted 24 May 2022
Available online 3 June 2022

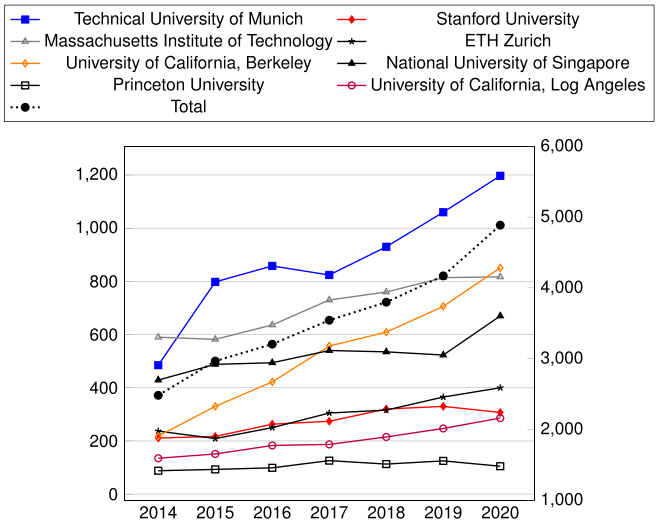

Fig. 1. The number of computer science degrees (bachelor's and master's) conferred per year by renowned universities in the area has steadily increased over the last decade. Data was collected from statistics published by the universities. The left y-axis represents the number of degrees per university. The right y-axis represents the **Total** number of degrees across all universities.

Multiple-choice quizzes are easy to assess, and automated tools are broadly available in learning management systems and paper-based assessments. However, mastery of these quizzes does not require problem-solving skills because they typically target only lower cognitive skills, particularly *knowledge* recall and *comprehension*. It is difficult to create quizzes that stimulate higher cognitive skills, such as problem-solving, essential in computer science (Alario-Hoyos et al., 2016; Williams & Haladyna, 1982).

Open-ended textual exercises enable instructors to teach problem-solving skills and allow students to improve their knowledge. These exercises do not have a single correct solution but rather allow answers within a particular solution space that words and phrases can characterize. Students profit from individual feedback relationships with their instructors (Feynman, 1994). Individual feedback and formative assessments are essential elements in learning (Higgins et al., 2002; Irons, 2007). Feedback on open-ended exercises allows students to try out problem-solving and experience failure. Students need guidance in the form of feedback in their learning activities to prevent misconceptions (Kirschner et al., 2006).

However, textual exercises lead to a wide answer spectrum because students need to formulate individual answers to problems, which results in an increased manual effort when reviewing students' answers. In addition, assuring consistent feedback is difficult with a large number of teaching assistants. This article describes a machine learning-based system as the solution to this problem.

This article is organized following *the design and engineering cycle* (Wieringa, 2014). Section 2 formulates the *design science research goals*, the *artifact design goal*, and *knowledge goals*, which we use to derive *knowledge questions* throughout the article. Section 3 describes grading efforts in large courses and the role of feedback in the learning process. Section 4 introduces the computer-aided feedback for textual exercises (CoFee) framework with its problem domain and dynamic behavior. Section 5 describes background literature and compares related work to CoFee. Section 6 validates the concepts of CoFee in a laboratory experiment. Section 7 describes the reference implementation Athena in the context of Artemis. Section 8 describes the course "Introduction to Software Engineering" in which the approach was used, shows the quasi-experimental study design of the empirical evaluation, presents results and limitations, and discusses the findings. Section 9 concludes the article with its main contributions, and Section 10 outlines future work.

2. Methodology

This research focuses on two main stakeholders: instructors, especially those responsible for large lecture courses, and students. For this paper, we define instructors as both lecturers and teaching assistants. Lecturers are university employees such as professors, researchers, and doctoral candidates. Teaching assistants are experienced students who have previously passed the same course with a good grade and are motivated to help in the teaching process. Some universities also use the term "tutor" to refer to a teaching assistant.

Lecturers have an interest in delivering high-quality teaching supported by many exercises. Through individual feedback, lecturers want to support students in their learning activities as much as possible. However, lecturers want to minimize their workload on assessments to have time to create and improve exercises and course materials. Teaching assistants need to balance their limited working hours between assessments, face-to-face teaching sessions, and answering questions. Students want to understand the course content, solve the exercises, and receive timely feedback. They want to re-iterate their solution based on feedback to fail early and learn from the mistakes on the way (Popper, 1934, 1959).

We focus on automating the assessment of textual exercises to meet the conflicting goals of producing high-quality feedback and saving time.

Research Goal: ***Reduce assessment efforts on textual exercises for instructors while scaling feedback for large courses.***

Following Wieringa's design science methodology (Johanβen, 2019; Wieringa, 2014), we break down this *research goal* into a goal hierarchy shown in Fig. 2. The *design science research goals* support the *social context goals*, which in turn are defined by the *external stakeholder goals* and the *problem context*. To achieve the research goal, we explore ways of automating and supporting the assessment process for textual exercises. Therefore, we conclude this with the following Artifact design goal:

Artifact Design Goal: ***Design a system that automatically assesses textual exercises.***

Section 4 describes the CoFee framework to generate computer-aided feedback for textual exercises. Section 7 describes a reference implementation for CoFee, the Athena software system. Athena collects assessment knowledge in the form of exercise and feedback pools. We summarize this effort to understand the stakeholders and the problem context with the following knowledge goal:

Knowledge Goal 1 (Investigation): ***Understand grading efforts and the role of feedback in large courses.***

Next, we want to validate if the proposed treatment, CoFee, is suited to solve the assessment problem for textual exercises. We address this with the second knowledge goal:

Knowledge Goal 2 (Validation): ***Understand the performance of CoFee and its individual components during the assessment of textual exercises.***

Last, we want to evaluate the implemented artifact, the Athena system, and analyze its performance in large courses. Therefore, we conclude with the third knowledge goal:

Knowledge Goal 3 (Evaluation): ***Understand the influence of Athena on the grading process.***

3. Problem investigation

3.1. Feedback in the learning process

There is clear evidence that guidance is essential to facilitate learning and prevent misconceptions (Kirschner et al., 2006). Therefore, it is important to involve students in learning activities, even in large courses. Examples and exercises play a central role in the early phases of cognitive skill acquisition (VanLehn, 1996). Carefully developed

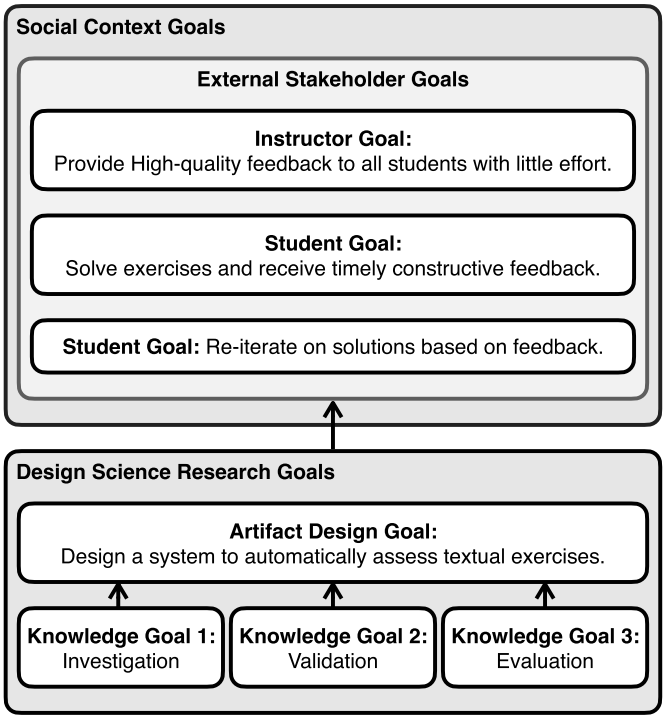

Fig. 2. Hierarchical goal taxonomy following the template from Wieringa (2014). An arrow indicates that a goal supports the other.

examples increase the learning outcome (Sweller & Cooper, 1985; Trafton & Reiser, 1993). Providing individual feedback is essential in learning to improve students' skills (Higgins et al., 2002). Feedback helps students to understand their learning progress and helps in the learning process through reflection. In the current teaching paradigm, students take "the active role of … seeking, interpreting and using feedback as part of their learning process" (Jensen et al., 2021). Good feedback incentivizes students to invest time in an exercise and rethink their solution. Participation in exercises with feedback has a positive effect on academic performance (Förster et al., 2018).

3.2. Interactive learning

Interactive learning is a scalable and adaptive teaching philosophy based on "constructive alignment" that puts the interaction with a student into the core of the educational activities (Krusche, Seitz, et al., 2017). It integrates aspects of team-based learning and creativity to stimulate problem-solving skills and soft skills.

Interactive learning decreases the cycle time between teaching a concept and practicing it during the lecture in multiple short iterations: Instructors teach and exercise small chunks of content in short cycles and provide immediate feedback so that students can reflect on the content and increase their knowledge incrementally. Interactive learning expects active participation of students and use of computers (laptops, tablets, or smartphones) in classrooms. Fig. 3 shows the iterative process of interactive learning, where each iteration consists of five phases that are performed several times during each lecture:

1. **Theory**: The instructor introduces a new concept and describes the theory behind it. Students listen and try to understand it.
2. **Example**: The instructor provides an example so that students can refer the theory to a concrete situation.
3. **Practice**: The instructor asks the students to apply the concept in a short exercise adapted to the individual student's existing knowledge and skills. The students submit their solutions to the exercise.
4. **Feedback**: The instructor provides immediate feedback to the student submissions using an automatic assessment system. Alternatively, the instructor can show multiple exemplary solutions and discuss their strengths and weaknesses.
5. **Reflection**: The instructor facilitates a discussion about the theory and the exercise to reflect on the first experience with the new concept.

3.3. Artemis

Artemis (Krusche & Seitz, 2018) is a teaching platform that supports interactive learning and is scalable to large courses with immediate and individual feedback. It is open-source[2] and used by multiple universities and courses.

Artemis includes several functionalities to implement interactive learning. In the following section, we present and discuss the essential features. Instructors can create different exercises: programming, modeling, quiz, text, and file upload. Artemis offers different assessment modes: automatic, semi-automatic, and manual. It automatically assesses programming and quiz exercises and provides a semi-automatic assessment approach based on machine learning for modeling and text exercises.

Artemis allows students to work collaboratively on the solution to the given tasks in team exercises. Instructors can incorporate live streams, recordings, and slides of lectures and embed exercises directly into them using lecture units. Students can ask questions and receive answers in a chat-based communication with emojis and references next to exercises and lectures. In addition, Artemis offers an exam mode for online exams. The exam mode includes additional functionalities, such as exercise variants, plagiarism checks, and offline support.

3.4. Assessment

Assessment is a time-intensive (Chen et al., 2018; Cheng, 2017), manual, and repetitive job. Efforts vary based on the size of the accepted answer space: Lower cognitive processes are easier to assess (e.g.,

[2] Artemis: https://github.com/ls1intum/Artemis.

remember) compared to higher processes (e.g., evaluate). That means if an answer asks to state a term, the assessment is simple as the answer either matches the solution or not. For complex exercises, students are free in their answers and, e.g., explain a concept based on an example. In this instance, assessment is difficult and time-consuming as graders need to analyze the example and solve the exercise in the students' context themselves. In software engineering, many solutions can be acceptable for a problem. Acceptable answers might change as paradigms shift, and new engineering principles become the norm.

To address *Knowledge Goal 1 (Investigation)*, we extrapolate the assessment efforts required for large courses following the interactive learning model to answer two knowledge questions (KQs):

Knowledge Question 1: ***How many assessments do large courses need?***

In the following, we calculate the required assessments for a course featuring three lecture exercises and four homework exercises every week. The course format is based our course "Introduction to Software Engineering" (ISE) (cf. Subsection 8.1). We assume 2000 participating students for one semester of 13 weeks:

$$\underline{\#exercises} = (3+4)\cdot 13 = \underline{91} \tag{1}$$

$$\underline{\#assessments} = \#exercises\cdot\#students = 91\cdot 2,000 = \underline{182,000} \tag{2}$$

We conclude that an interactive course sets 91 exercises over the course of the semester. Therefore, instructors need to complete 182,000 assessments in a large course with 2,000 students (KQ 1).

Knowledge Question 2: ***How much time do instructors spend on manual assessments of exercises?***

Given an average assessment time of 5 min per student solution, we extrapolate the assessment total assessment time:

$$\underline{\Sigma\ Assessment\ Time} = \#assessments\cdot 5\ min \tag{4}$$

$$= 15,166.\overline{6}h = \underline{1,166.\overline{6}h/Week} \tag{5}$$

We conclude that the large course requires 15,167 h of assessment work which translates to 1,167 h every week (KQ 2). Data from ISE in 2021 shows that out of a total of 89 exercises, 24 were textual exercises (27%). We, therefore, estimate that instructors need to spend 315 h on assessments every week for textual exercises alone.

4. Treatment design – CoFee

To address the artifact design goal stated in Section 2, we derive an artifact design problem, which we define by following the template proposed by Wieringa (2014): We highlight artifacts, requirements, and stakeholder goals.

We investigate how to provide students with feedback on their exercise solutions automatically. We present the CoFee approach, which captures knowledge during the assessment process and provides instructors with feedback suggestions. CoFee allows instructors to assess exercises faster and offer consistent feedback to students in large courses. We summarize this as follows:

Artifact Design Problem: ***How to implement a system (artifact) that generates feedback on textual exercise solutions (requirement) so that instructors can give better feedback in shorter cycles (stakeholder goal)?***

This section introduces the proposed treatment computer-aided feedback for textual exercises. We describe the architecture and dynamic behavior of the treatment CoFee.

4.1. Architecture

Fig. 4 depicts the analysis object model (Bruegge & Dutoit, 2009) of the problem domain: A Course consists of many Exercises. Students can participate in an exercise by submitting their solutions. A Submission

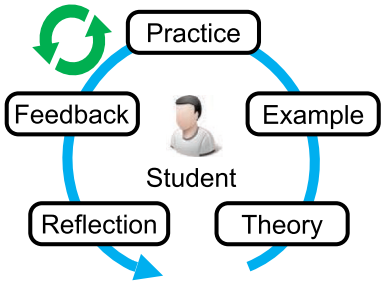

Fig. 3. Interactive learning puts the individual student into the core of the learning activity and follows an iterative process that is conducted multiple times in lectures.

can be decomposed into many Segments. Each of them encapsulates one core idea of the answer. Segments can receive Feedback via a comment and a score. We model the automatic generation of feedback and instructor grading as a metaphorical factory, following the *factory method pattern* (Gamma et al., 1994). Following the metaphorical application of design patterns, the instructor is an expensive feedback source. To cautiously and efficiently use this expensive subject, we introduce the Automatic feedback engine as a proxy object to filter which feedback requests it needs to forward to the *real subject*, the Instructor.

4.2. Dynamic behavior

Fig. 5 presents an overview of the workflow. CoFee first segments a submitted answer by splitting the answer into topically-coherent segments. These segments are annotated with one or more feedbacks as they cover a single core idea. Next, CoFee groups the segments into clusters by the similarity of their ideas. Based on the cluster classification, CoFee suggests gradings based on the assessment knowledge from the feedback pool. If enough assessment knowledge has been collected for a specific segment, then automatic feedback can be suggested. Otherwise, an instructor is required to complete the grading. Finally, CoFee presents a partial grading to allow the instructors to benefit from the knowledge generated. Instructors accept, change, or discard existing feedback suggestions and provide new feedback. All feedback is submitted to the feedback pool for reuse in future grading sessions.

CoFee learns which answers to an exercise are considered correct in the learning context. For further submissions, the learning platform automatically generates suggestions for similar answers or even automatically evaluates the answers. In doing so, the learning platform uses the knowledge of previous assessments from lecturers. The more students participate in an exercise, the more knowledge is generated and the better feedback the learning platform can suggest.

This addresses the *external stakeholder goals* stated in Section 2. The instructor's goal is to provide high-quality feedback to all students while decreasing the overall assessment time. The student's goal is to receive timely feedback. CoFee integrates into existing learning platforms that need to provide an interface for students to submit their textual answers. We utilize a segment-based feedback concept (Bernius & Bruegge, 2019), requiring assessors to provide feedback and score about a segment of a student's answer, resulting in relatable and reusable feedback elements.

CoFee trains its assessment model with every feedback element and becomes more accurate with every new feedback element. After the assessment process, the system can detect conflicting assessments in both comments and scores. Therefore, CoFee computes the similarity among feedback comments. We claim that the similarity between two segments should be proportional to the similarity between the feedback comments. If this relation is violated, CoFee prompts the instructor to

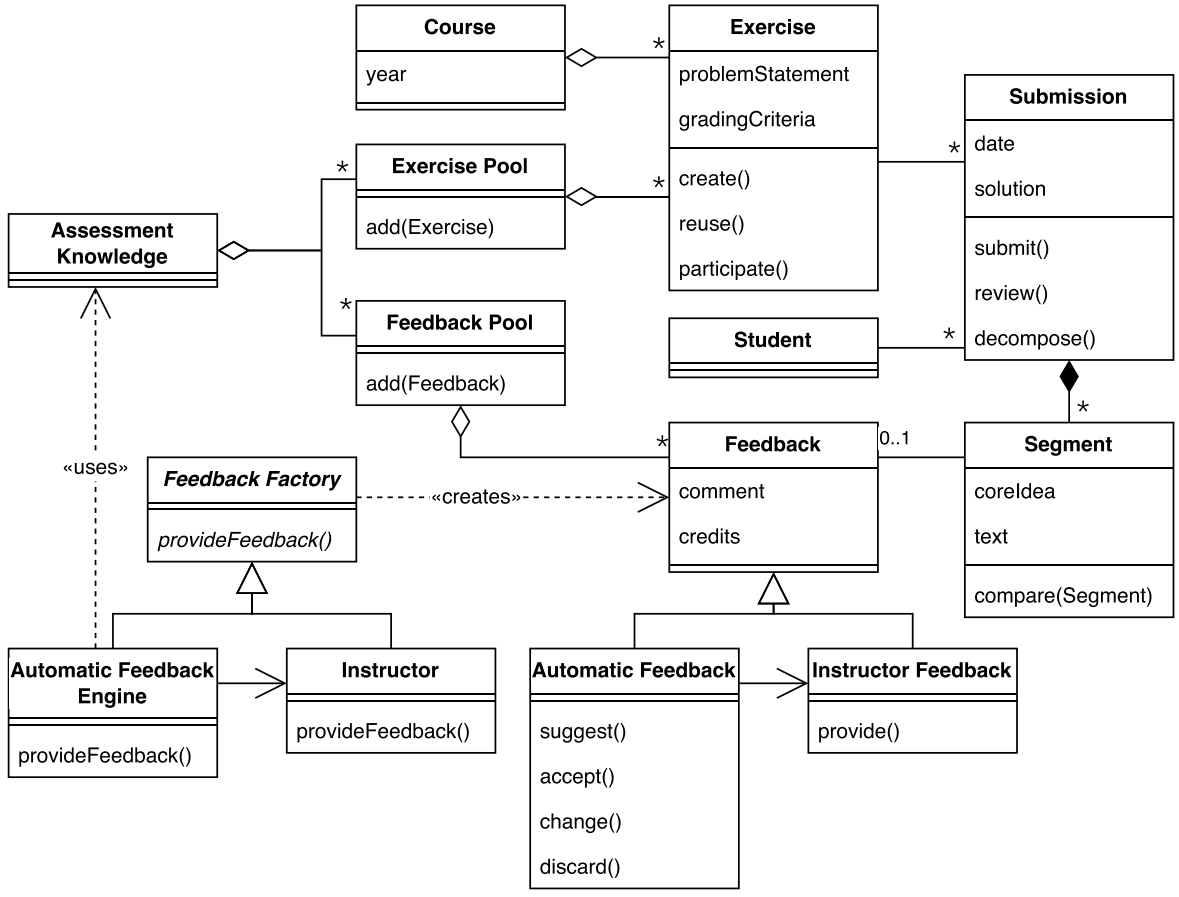

Fig. 4. Analysis Object Model of the CoFee framework. The model describes the system from the stakeholder's point of view and illustrates the concepts visible to the stakeholder (Bruegge & Dutoit, 2009) (UML Class Diagram).

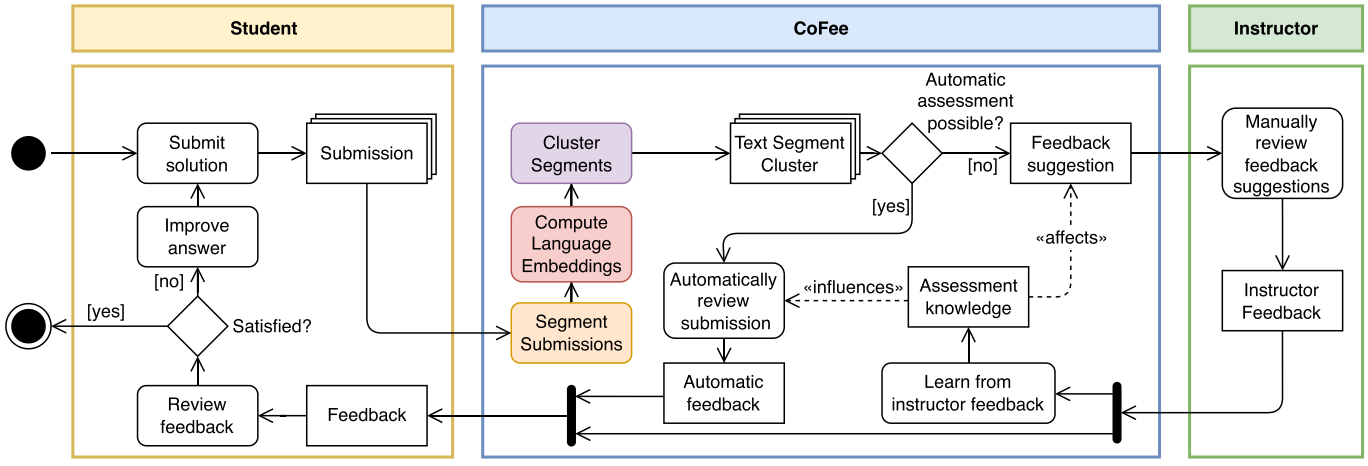

Fig. 5. Workflow of automatic assessment of submissions to textual exercises based on the manual feedback of instructors. CoFee analyzes manual assessments and generates knowledge for the suggestion of computer-aided (automatic) feedback (UML activity diagram).

review the pair of submissions and allows them to update the assessment as needed. The learning platform may only release the feedback to students after the instructors have resolved inconsistencies.

5. Related work

This section compares CoFee to alternative treatments from related work in the literature. Compared to existing work, CoFee segments and clusters student solutions automatically. By training the system during the assessment process, we do away with a reference dataset before the assessment. Furthermore, by training with correct and incorrect solutions, we maintain a dataset to provide helpful feedback comments to support the learning process. Finally, dynamically collecting the dataset during assessment keeps the system independent of any domain and allows for using the system with new exercises to incorporate the latest knowledge into teaching.

5.1. Assessment systems

Automated essay scoring computes scores on written solutions based on previous submissions. Automated essay scoring systems require a perfect solution to be available upfront (Mitchell et al., 2002; Pulman & Sukkarieh, 2005; Sukkarieh et al., 2003). They primarily consider the similarity to a perfect solution to determine the grade. Giving feedback is not the focus of automated essay scoring systems. Manual clustering and shared grading are concepts used in research (Pérez et al., 2005) and commercial tools (i.e., Gradescope). Managing clusters is hard at scale, especially communicating the exact differences between clusters among many graders.

5.1.1. Atenea

Atenea is a computer-assisted assessment system for scoring short answers in computer science (Pérez et al., 2005). Atenea maintains a database of short-answer-questions with corresponding sample solutions. Sample solutions are either written by an instructor or reused from a highly graded student answer. Atenea combines latent semantic analysis (LSA) and a modified bilingual evaluation understudy algorithm hypothesizing that syntax and semantics complement each other naturally. Combining these two natural language processing (NLP) tools always performs better (with a higher hit rate). Furthermore, Pérez et al. (2005) argue that syntactical and semantical analysis combinations lead to greater automatic text assessment results.

Atenea compares student answers to a set of predefined answers. It determines a grade based on the similarity to these predefined answers. This approach is limited to exercises with a narrow answer space where possible answers are known beforehand. High variability in answers limits Ateneas applicability and requires a large set of sample solutions. The focus of the Atenea system is grading, whereas Athena primarily focuses on individual feedback. Athena does not require a sample solution but collects knowledge on correct and incorrect solutions during the manual assessment. The evaluation of Atenea focuses on comparing NLP techniques in the context of grading using a dataset. We evaluate Athena by using it in multiple courses and measuring its performance.

Atenea compares student answers to a set of predefined answers. Its similarity to these predefined answers determines the grade. This approach is limited to exercises with a narrow answer space where possible answers are known beforehand. High variability in answers requires a large set of predefined answers, limiting the system's applicability. The focus of the Atenea system is grading, whereas Athena is primarily focused on individual feedback. Athena does not require a predefined solution but collects knowledge on correct and incorrect solutions during the manual assessment. The evaluation of the Atenea authors focuses on a comparison of NLP techniques in the grading context and is based on a dataset. We evaluate Athena by using it in multiple courses and measuring its performance.

5.1.2. Powergrading

Powergrading is an automatic assessment approach for textual exercises (Basu et al., 2013) that provides feedback in the form of a numerical score and a comment explaining why an answer is correct or incorrect, similar to the comment of a human. In addition, Basu et al. (2013) propose a system that clusters similar answers to a question so that instructors can "divide and conquer" the correction process by assessing a whole cluster with the same score and comment, therefore reducing the correction time significantly. Clustering answers to a question should happen based on a distance function composed of different features and automatically tries to learn a similarity metric between two students' answers. Some of the implemented and used features that are weighted in developing this distance function used for clustering are, e.g., the difference in length between two answers, the term frequency-inverse document frequency (TF-IDF)[3] similarity of words, or the LSA vectorial score based on the entirety of Wikipedia as a training text corpus. The authors have tested their implementation with test data from the United States Citizenship Exam in 2012 with 697 examinees. They concluded that around 97% of all submissions can be grouped into similar clusters so that instructors would only have to provide feedback for a single cluster and would still be able to reach and correct multiple submissions at once, therefore reducing assessment time significantly (Basu et al., 2013).

Powergrading is focused on short-answer grading, where a typical answer does not exceed two sentences. Athena is not limited to a certain answer length and uses segmentation to work with multiple sentences or paragraphs. Similar to Powergrading, Athena groups segments into clusters. Both systems assume hierarchical cluster structures. Powergrading allows instructors to grade clusters rather than submissions, whereas Athena will use the cluster structure to suggest feedback for the following assessments.

5.1.3. Gradescope

Gradescope[4] is a system geared toward assessing handwritten homework and exam exercises (Singh et al., 2017) by scanning paper-based work. Instructors grade the submissions online. Gradescope allows the instructor to create grading rubrics at the assessment time dynamically. Instructors can group similar submissions manually for shared grading or rely on suggested groups for the assessment.

Athena also provides sharing feedback with groups of answers; however, Athena groups individual segments, whereas Gradescope groups entire submissions. Gradescope allows the grader to grade multiple submissions as one, similar to Powergrading, whereas Athena shares individual feedback elements across multiple submissions. Athena requires instructors to inspect every submission and supports instructors by suggesting feedback items. Neither system requires a training dataset of previously assessed answers. For exercises with a limited answer spectrum, Gradescope does allow the grader to assess several submissions efficiently as it reduces the number of solutions to grade. However, this approach is more limited for exercises with high variability in answers (e.g., when asking for examples) as more groups with fewer elements need to be graded.

5.2. Language models

Automatically assessing text submissions requires comparing segments of those submissions and identifying similar pieces of text. Therefore, we need a measurable abstraction of a text's meaning as an intermediate representation. This paper relies on existing approaches and techniques from the domain of NLP, most notably language models and word embeddings, to convert a piece of text into a comparable format. Student answers can contain unknown words, incorrect grammar and punctuation, and false statements.

Word embedding is a feature learning technique in NLP, where words or phrases from the vocabulary are mapped to vectors of real numbers (each word is associated with a point in a vector space) (Li & Yang, 2018). The feature vector represents different aspects of the word, and consequently, words with the same meaning are assigned similar vector representations. Additionally, word embeddings can capture word analogies by examining various dimensions of the differences between word vectors (Pennington et al., 2014). For example, the analogy "king is to queen as man is to woman" should be encoded in the vector space by the vector equation $king - queen = man - woman$.

The distributed representation is learned based on the usage of the

[3] TF-IDF: An information extraction statistic that indicates how significant a word is to a document (Ramos, 2003).

[4] Gradescope: https://gradescope.com.

Fig. 6. 760 text segments clustered by hand into 75 clusters.

words. This allows words used in similar contexts to have similar representations, naturally capturing their meaning. Embeddings from Language Models (ELMo) (Peters et al., 2018) is a word embedding constructed as a task-specific combination of the intermediate layer representations in a bidirectional language model. It models complex characteristics of words-use in the language dictated by the syntax and semantics. It also captures how these uses vary across linguistic contexts, which is important for addressing polysemy in natural languages.

In a deep language model, the higher-level long short term memory states are shown to capture context-dependent aspects of word meaning while lower-level states model aspects of the syntax. By constructing a representation out of all the layers of the language model, ELMo can capture both language characteristics. ELMo representations have three main characteristics to achieve state-of-the-art results in most common NLP downstream tasks. First, ELMo representations are contextual: the representation for each word depends on the entire context in which it is used. They are also deep: the word representations combine all layers of a deep, pre-trained language model neural network. Finally, ELMo representations are purely character-based, allowing the network to use morphological clues to form robust representations for out-of-vocabulary tokens, unseen in training.

6. Treatment validation

We validate the treatment using a laboratory experiment to study the feasibility of CoFee. The treatment validation answers two knowledge questions that address the effects of the treatment artifacts:

Knowledge Question 3: ***Do groups of similar segments occur which can receive the same feedback?***

Knowledge Question 4: ***What portion of solutions can CoFee assess?***

Answering Knowledge Questions 3 and 4 addresses Knowledge Goal 2.

6.1. Exercise

We collected a dataset by running a textual exercise in the "Project Organization and Management" (POM) course at Technical University of Munich (TUM) using the Artemis platform. In the exercise "iterative vs. incremental vs. adaptive," students were asked to differentiate the terms *iterative development*, *incremental development*, and *adaptive development* using examples. 130 students participated in the exercise.

6.2. Study design

We manually evaluated all submissions by segmenting the answers and separating all segments by their core idea. The 130 student submissions resulted in 762 text blocks. We printed all segments on paper cards and manually clustered them into groups by similarity in several iterations. Fig. 6 shows the paper cards with classifications marked using sticky notes. In the first iteration, we roughly sorted them into three clusters. We then continued to subdivide each cluster in the following iterations. The similarity refinement increased with every iteration over the whole data set. We repeated the process until we reached a satisfactory assignment into 75 clusters.

6.3. Results and findings

We identified that 95% of all segments could be assigned to clusters. We found a total of 66 clusters in the dataset. The average cluster has 11 elements with a minimum of two and a maximum of 49; the median cluster size was four. 717 out of the 762 segments can be assigned to a cluster (94%).

Finding 1 (Clusters): ***Clustering of segments for shared grading is possible. The majority of segments*** **(94%)** ***can be clustered.***

The experiment results show that student solutions can be split into

segments and grouped by similarity. Furthermore, the data suggests that 94% of segments can be part of a grading cluster.

Finding 2 (Grading Potential): ***Grading efforts can be reduced by 85% through automatic grading of clusters.***

The overlap between student answers can reduce grading to one segment per cluster and unclustered segments. In this instance, the grading can be reduced to 85%:

$$(717 - 66)/762 = 85.4\% \tag{7}$$

The unclustered portion of 6% is not suitable for the concept of shared grading. Therefore, the overall grading effort is reduced from 762 segments to 111 segments, which is 15% of the original grading effort:

$$(66 + 45)/762 = 14.5\% \tag{8}$$

With these findings, we conclude that CoFee is a suitable treatment for the *artifact design goal*, and we proceed with the treatment implementation.

7. Treatment implementation – Athena

We implemented CoFee in a reference implementation called Athena[5] (Bernius et al., 2021) integrated into the learning platform Artemis (Krusche & Seitz, 2018). After the exercise deadline, Artemis sends the students' answers to Athena for processing. Athena will preprocess the answers before the assessment begins and identify segments suitable for the same feedback. Fig. 7 depicts the preprocessing activities: The system analyzes incoming student answers using NLP, divides them into text segments, and uses them to create text clusters with similar text segments from different answers. This is done using a combination of segmentation and linguistic embeddings, particularly deeply contextualized word representations (i.e., ELMo). This allows for an understanding of students' responses and the generation of individualized feedback. In this way, a learning platform can automatically reuse manual feedback for contributions from different students. Automatic individualized feedback suggestions can reduce the workload for instructors and increase the consistency and quality of feedback to improve students' understanding. Fig. 8 depicts the top-level design of the system, which consists of three steps: segmentation, language embedding, and clustering.

First, Athena analyzes the answers (incoming text) to identify segments (Bernius et al., 2020). Therefore, Athena identifies common topics described in the answers from all students. A keyword represents a topic. To identify the important topics for an exercise, Athena counts the occurrences of lemmatized words across all students and selects the ten most common words (Bernius et al., 2020). Next, Athena will break down every student's answer into clauses. Adjacent clauses that share the same topic, represented by a keyword and the absence of a new keyword, are merged to form a segment. If a new keyword appears in the following clause, we identify a topic shift and start a new segment. The result is a set of topically coherent segments.

Second, Athena uses an ELMo model to convert each segment to vector form. ELMo vectors have 1,024 dimensions representing the information extracted from the segment. The vector representation allows for a comparison of segments and identifying similarities. Athena uses a pre-trained ELMo model (Peters et al., 2018) based on a dataset consisting of 5.5B tokens from Wikipedia and news articles.[6]

Third, Athena employs the Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) clustering algorithm (McInnes & Healy, 2017) to identify classes of similar text segments. Within a cluster, Athena shares manually created feedback as suggestions. The hierarchical clustering algorithm allows to determine the number of clusters dynamically. Further, the hierarchical structure can dynamically narrow or widen the search radius depending on the availability of feedback. Narrow clusters provide more accurate feedback on the one side; but also limit the possible coverage. Larger clusters increase the possibility of finding existing feedback to compose a suggestion; however, they also increase the risk of false feedback.

During the manual assessment, Athena uses a prioritized assessment order. Submissions with several segments in clusters without feedback are prioritized, maximizing the possible coverage for automatic feedback suggestions. Athena searches their respective clusters for each segment for existing feedback and suggests the closest feedback. Furthermore, credit points associated with feedback are prioritized based on the clusters' credit average. Athena's automatic feedback suggestions are displayed to instructors within Artemis as part of the assessment interface (Bernius & Bruegge, 2019), as depicted in Fig. 9. Instructors can add additional feedback to unassessed parts of the student solution. They can either approve the feedback suggestions or update them as they see fit.

8. Implementation evaluation

This evaluation compares the quantity and quality of feedback in the course ISE with and without the Athena system. We analyze feedback in three instances of ISE: In 2019, text exercises on Artemis were introduced during the course. The Artemis platform served as the submission and feedback platform for students. All feedback was composed manually and published through Artemis. In 2020, the course introduced the Athena system as part of Artemis. Students continued to use Artemis to submit their feedback. Instructors receive feedback suggestions from Athena when reviewing student answers. Instructors need to check the feedback suggestions, add additional feedback where needed, and can also update feedback suggestions as needed. In 2021, the course continued its use of the Athena system. As part of this experiment, tutors needed to manually review exercises during the first half of the course. The Athena system was enabled for the second half, and tutors had to work with the suggested feedback.

We compare the feedback for exercises using the Athena system (*treatment*) with feedback composed manually (*control group*). We compare the quantity of feedback, the quality of feedback comments, the student satisfaction, and the assessment efforts before and after introducing the Athena system, the introduced intervention.

In this section, we describe the course ISE and the study design of the evaluation. The evaluation consists of two parts. The first part analyzes the feedback generated by the system Athena. We analyze how many assessments receive feedback from Athena by inspecting exercises from 2020 to 2021 where the system was used. This can be summarized in the following knowledge question:

Knowledge Question 5: ***What portion of grading can be supported by Athena?***

Further, we study the quality of the feedback suggestions. Therefore, we study how instructors interact with the suggested feedback. Finally, as instructors can overwrite the feedback suggestions, we analyze how much feedback is published to the students. We summarize this as follows:

Knowledge Question 6: ***How accurate is Athena feedback?***

The second part of the evaluation compares Athena feedback to instructor feedback. Therefore, we first ask students to rate their feedback and compare how Athena feedback performs compared to instructor feedback.

Knowledge Question 7: ***How do students perceive Athena feedback?***

Second, we analyze student complaints on their feedback to study if Athena feedback has a higher quality and attracts fewer complaints than instructor feedback.

Knowledge Question 8: ***Does Athena feedback reduce the number of student complaints on their feedback?***

[5] Athena: https://github.com/ls1intum/Athena.

[6] AllenNLP - ELMo: https://allennlp.org/elmo.

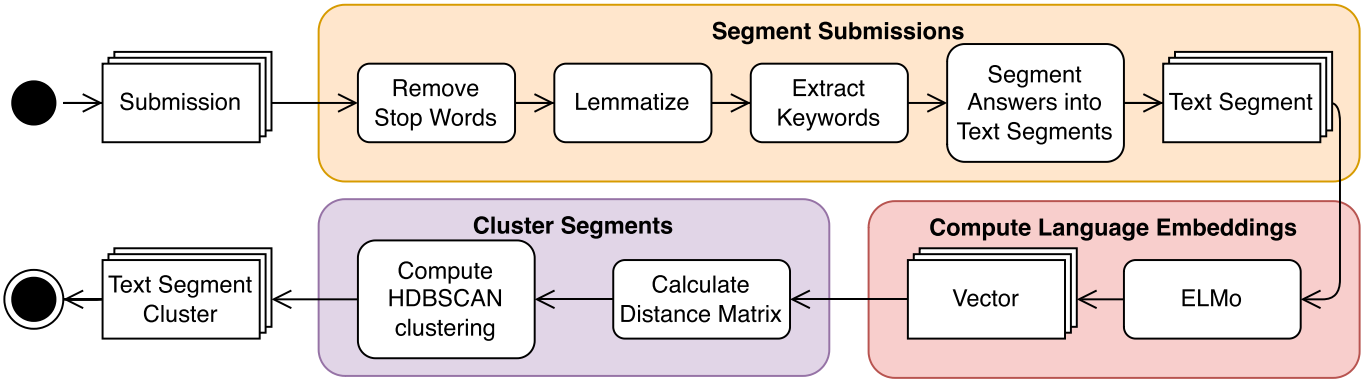

Fig. 7. Overview of the machine learning activities making up the "Segment Submissions", "Compute Language Embeddings", and "Cluster Segments" activities in Fig. 5. These are used to extract text segments and build text clusters for scoring and similarity analysis (UML activity diagram).

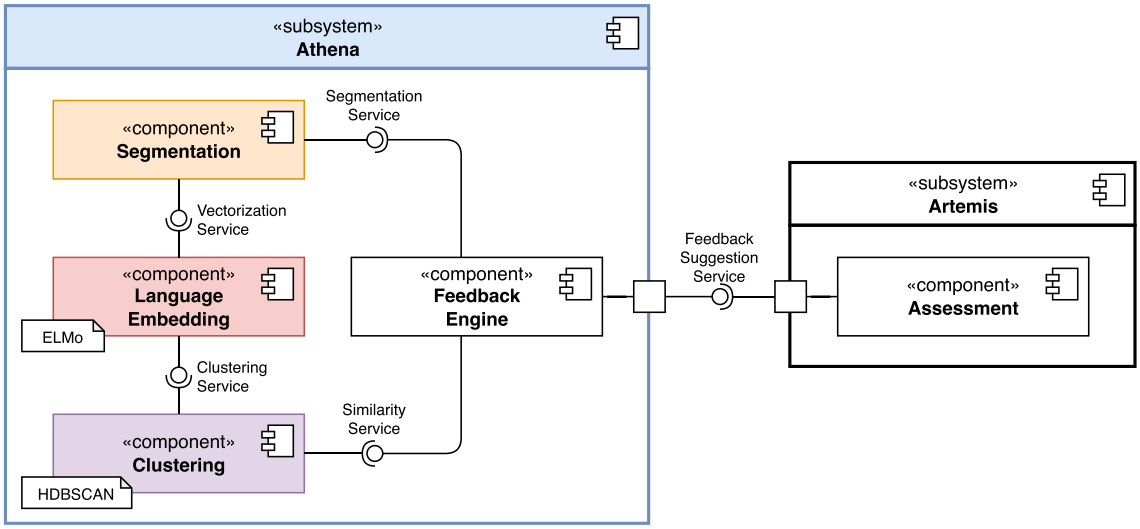

Fig. 8. Top-level design of the Athena system. Athena is composed of four components: Segmentation, Language Embedding, and Clustering implement the machine learning activities depicted in Fig. 7. The Feedback Engine acts as the facade to Artemis and offers an API that the Assessment component uses to receive feedback suggestions.

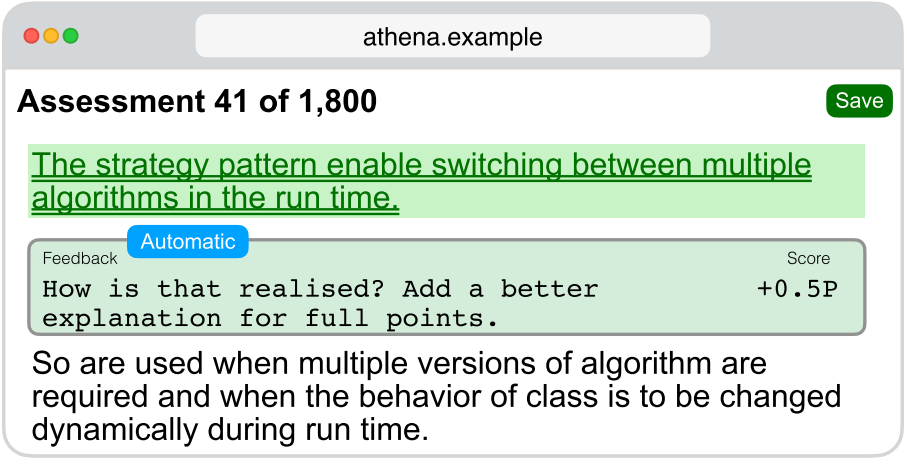

Fig. 9. Example of the instructor interface: Athena presents a feedback suggestion for the first text segment with a feedback comment and a score.

Third, we inspect if the semi-automatic assessment concept influences the quantity of feedback provided to students.

Knowledge Question 9: ***Does Athena generate more feedback than instructors?***

8.1. Course

The course ISE is an introductory software engineering course, with around 2,000 registered students who are mainly computer science bachelor's students in their second semester. Students with computer science as a minor can also enroll in the course. The course covers software engineering concepts, such as requirements analysis, system and object design, testing, lifecycles, configuration management, project management, and UML modeling (Krusche et al., 2020). Before starting the course, students need fundamental programming experience (e.g., Introduction to Computer Science or Fundamentals of Programming). The instructors use constructive alignment (Biggs, 2003) to align the teaching concepts and exercises with the course objectives. For each lecture, they define learning goals based on six cognitive processes in Bloom's revised taxonomy (Anderson et al., 2001). The course focuses on higher cognitive processes: Students apply the concepts in concrete exercises.

Following an interactive learning approach, ISE teaches software engineering concepts with multiple small iterations of theory, example, exercise, solution, and reflection (Krusche & Seitz, 2019). Therefore, it utilizes exercises to foster student participation (Krusche, Seitz, et al., 2017) and motivate the students to attend the lectures (Krusche, von Frankenberg, & Afifi, 2017). The course involves different kinds of exercises:

1. Lecture exercises as part of the lectures
2. Group exercises solved in small ad hoc groups

Table 1
ISE exercises over the years with their Feed back Factory (cf. Fig. 4) used each year: Instructor Feedback (I), Athena Feedback (A), or Exercise not used (–).

Nr.	Exercise	2019	2020	2021
H.1	Text Exercise Tutorial	I	I	–
H.2	Different Models in SE	I	I	–
H.3	Group vs. Team	–	–	I
H.4	Change in Software Development	I	I	I
H.5	Purpose of Modeling	I	–	–
H.6	Model & View & System	I	–	–
H.7	Bumpers Nonfunctional Requirements	I	I	I
H.8	Difference Aggregation & Composition	I	I	I
H.9	Visionary Scenario for Bumpers	I	I	I
H.10	As-Is Scenario for Bumpers	I	I	I
H.11	**Coupling & Cohesion**	I	**A**	I
H.12	**Analysis Models & System Design**	I	**A**	I
H.13	Design Goals in Closed Architectures	I	–	–
H.14	**Design Goal Trade-offs**	I	**A**	I
H.15	Create a Formalized Scenario	–	–	I
H.16	Closed vs. Open Architecture	I	–	–
H.17	**Centralised vs. Decentralised Designs**	I	**A**	I
H.18	**Inheritance vs. Delegation**	I	**A**	**A**
H.19	Specification & Implementation Inheritance	I	I	I
H.20	**MVC & Observer Pattern**	I	**A**	**A**
H.21	**Advantages and Disadvantages of Scrum**	I	**A**	I
H.22	**Unified Process and Scrum**	I	**A**	–
H.23	**Spiral Model and Scrum**	–	–	**A**
H.24	**Problems using Git**	I	**A**	**A**
H.25	**Merge Conflicts & Best Practices**	I	**A**	**A**
H.26	Strategy vs. Bridge Pattern	I	–	–
H.27	Model Refactoring	–	–	I

3. **Homework exercises** to be solved throughout the week individually
4. Team exercises to be solved in a team in five 2-week periods
5. **Exam exercises** to assess the students' knowledge after the course has finished in multiple variants

Students were asked to submit their solutions to all but group exercises to Artemis to receive an assessment with feedback and points. The students could gain bonus points for the final exam when participating in the exercises. The instructors utilize programming, modeling, **textual**, and quiz exercises in the course to train software engineering and problem-solving skills. Table 1 lists all homework exercises conducted in the course and marks whether Athena Feedback was employed in 2019–2021.

8.2. Study design

Fig. 10 shows the study design of the evaluation that was instantiated for each exercise in which Athena was used for grading. The instructor defines the exercise in Artemis with a problem statement, grading criteria, example solutions, and a due date. The students can insert their solutions in plain text on Artemis. After the due date, Artemis sends all student answers to Athena to preprocess the answers as described in Section 7. The instructors can review the student answers as soon as Athena completes the preparation and stores the text clusters. The instructors create a review for every student's answer consisting of multiple feedback items. The instructors used a chat room during the review phase to discuss the grading criteria as needed.

Every review can either be classified into one of two categories: Instructor and Athena feedback. A review is considered to receive Athena feedback if at least one feedback item was suggested by Athena. Reviews without feedback suggestions receive Instructor feedback. Furthermore, Athena stores intermediate versions of all feedback items to evaluate how instructors work with feedback suggestions.

After the instructors completed the review, we retrieved the classification of the reviews from the Artemis database using SQL queries. Two researchers verified the correctness of the queries. We collected the statistics on the feedback items from Athena. We inserted the measurements in a spreadsheet for further analysis and graphing. Two researchers reviewed the results for consistency and plausibility and took several samples to check individual feedback entries.

8.3. Results: Athena Feedback

In the implementation evaluation, we answer five knowledge questions that address the influence of Athena on the grading process. These knowledge questions address knowledge goal 3 stated in Section 2.

First, we classify the reviews into two two categories. Fig. 11 and Fig. 12 depict the classification of the reviews. On average, 45% (Homework 25.2%, Exams 53.9%) of all reviews received Athena Feedback. In exercise *E.19*, the system performed best with 75% Athena feedback. Exercises *E.04* and *E.14* have the least coverage, with 6% Athena feedback.

Finding 3 (Coverage): ***Coverage Athena can cover up to 75% of reviews with feedback suggestions without previous training data or a predefined solution.***

Second, we further analyze the reviews classified to receive *Athena*

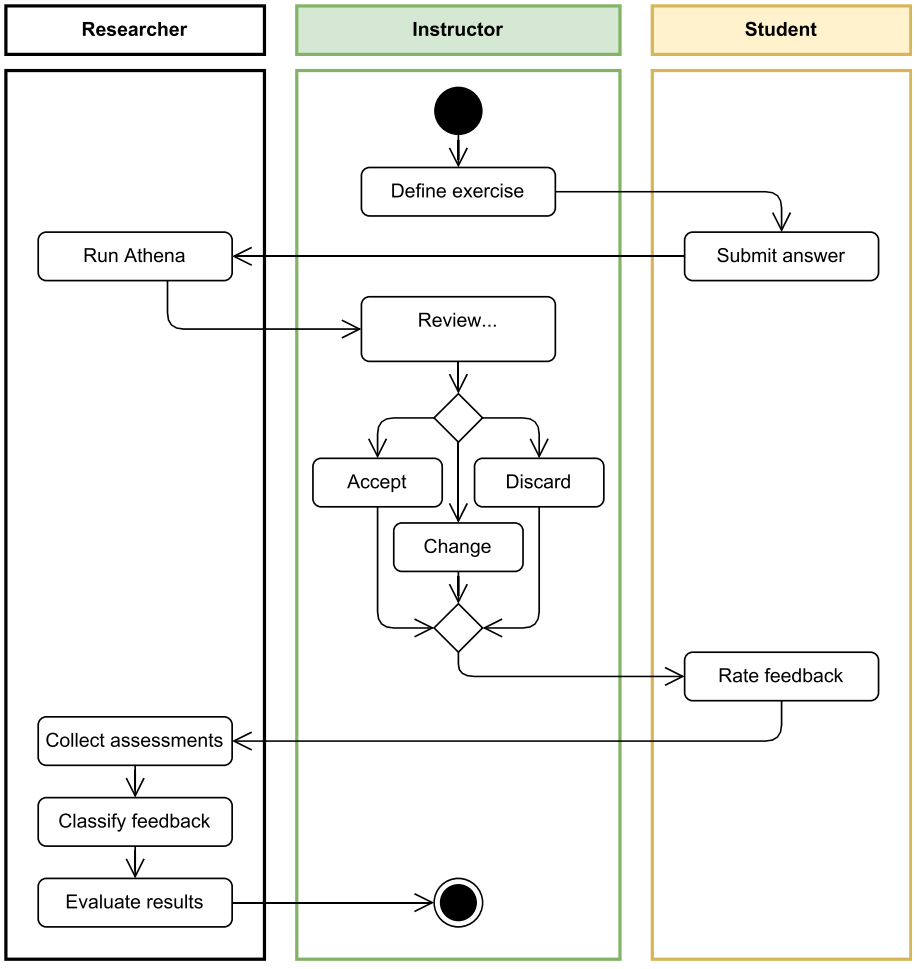

Fig. 10. Research approach depicted with the involved actors and flow of events (UML activity diagram).

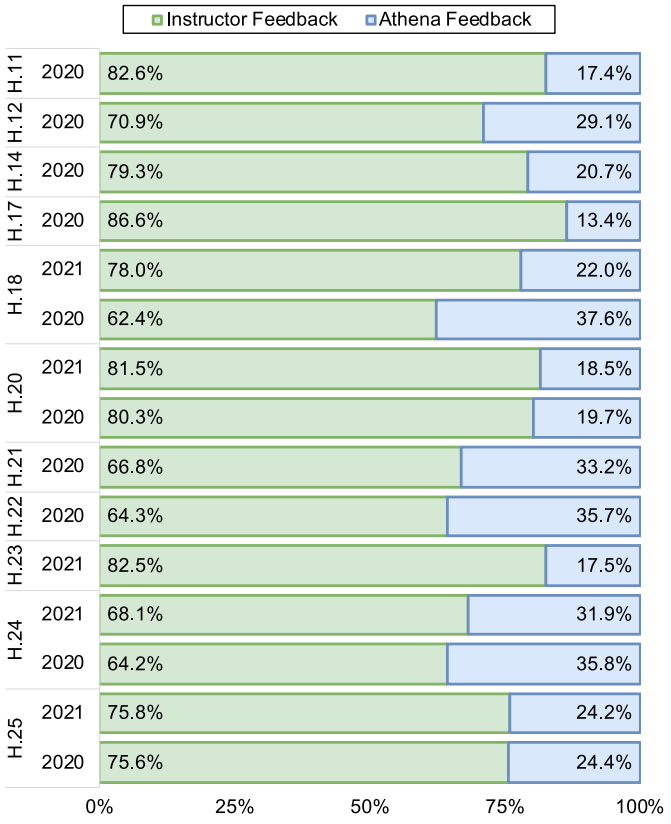

Fig. 11. Homework exercises with their assessment ratios. Athena feedback reviews received automated suggestions which were reviewed by instructors. On average, 25% of all homework assessments were computer-aided.

feedback above. Therefore, we inspect all feedback that is part of these reviews.

We formulate this data as a binary classification to evaluate Athena's performance. Feedback suggestions generated by Athena are *Positive*, and the absence of feedback for a given segment is *Negative*. We compare initial suggested feedback with the final feedback from the instructor (cf. Subsection 4.2) to classify both positive and negative suggestions as either correct (*True*) or incorrect (*False*). This leads to the following four classifications:

TP is a *True Positive* classification, in which Athena generated feedback on a segment that instructors published to students unmodified or slightly modified, e.g., with an extension.
TN is a *True Negative* classification, in which Athena did not provide feedback to a segment. The instructor did not see any need to providing feedback, either.
FP is a *False Positive* classification, in which Athena generated false feedback, and because of that, the instructor had to change the feedback.
FN is a *False Negative* classification, in which Athena did not suggest any feedback; however, feedback was needed for this segment. Therefore, the instructors had to intervene and compose their own feedback manually.

Following this classification, we can describe the performance of Athena following both the sensitivity and specificity values, as well as the accuracy (Witten et al., 2011):

The *recall* describes how much feedback has been correctly generated by the Athena system; this metric is also known as the sensitivity or the true positive rate (TPR).

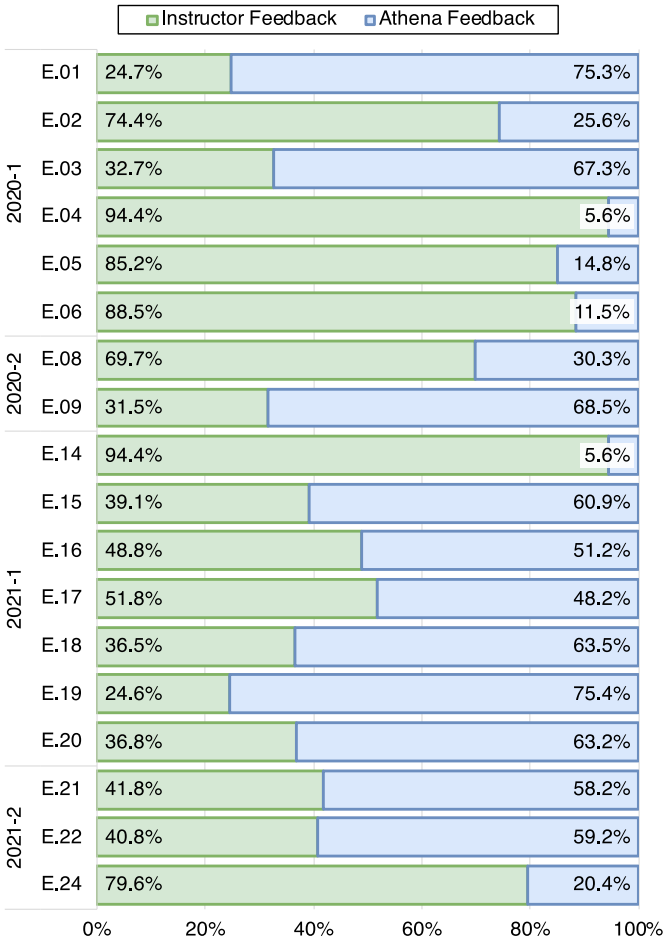

Fig. 12. Exam exercises with their assessment ratios. Athena feedback reviews received automated suggestions which were reviewed by instructors. On average, 54% of all exam assessments were computer-aided.

$$TPR = \frac{TP}{TP + TN} \tag{9}$$

The *specificity* describes the number of segments for which instructors did not provide feedback and were left without feedback by Athena; this metric is also known as the true negative rate (TNR).

$$TNR = \frac{TN}{TN + FP} \tag{10}$$

The *precision* describes the proportion of suggested feedback by Athena published to students by instructors; this metric is also known as the positive predictive value (PPV).

$$PPV = \frac{TP}{TP + FP} \tag{11}$$

The accuracy summarizes how much feedback was suggested and how many segments stayed without feedback correctly.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + TF} \tag{12}$$

Table 2 summarizes the binary classification results, which are visualized in Fig. 13.

Finding 4 (Precision): ***Athena augments instructor feedback precisely in most cases*** ($\overline{PPV} = 92\%$).

The precision of 92% confirms our previous findings (Bernius et al., 2021). This means that Athena Feedback successfully augments instructor feedback, rarely suggests incorrect feedback, and is appropriate and respects the context within the solutions. In addition, we

Table 2
Results of the binary classification (lest) and analysis (right).

Exercise	Year	TP	TN	FP	FN	TPR	TNR	PPV	Accuracy	F-score
H.20	2021	208	792	0	283	20.8%	100.0%	100.0%	77.9%	56.8%
H.21	2020	534	1727	66	1704	23.6%	96.3%	89.0%	56.1%	57.3%
H.22	2020	437	1108	41	887	28.3%	96.4%	91.4%	62.5%	63.2%
H.23	2021	198	678	13	369	22.6%	98.1%	93.8%	69.6%	57.6%
H.24	2020	525	868	35	1279	37.7%	96.1%	93.8%	51.5%	72.3%
	2021	416	959	49	1097	30.3%	95.1%	89.5%	54.5%	64.3%
H.25	2020	265	722	23	444	26.8%	96.9%	92.0%	67.9%	61.9%
	2021	291	1007	39	582	22.4%	96.3%	88.2%	67.6%	55.6%
Overall		2874	7861	266	6645	26.6%	96.9%	92.2%	63.5%	61.1%

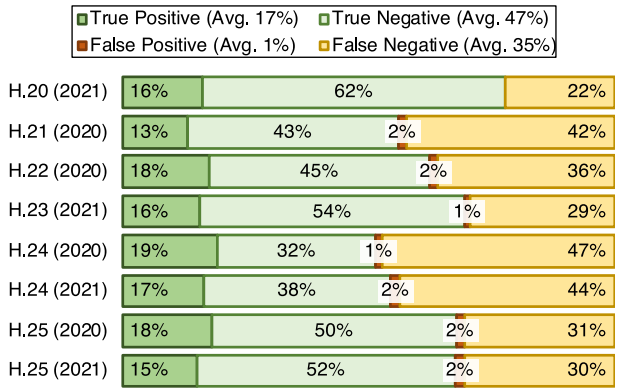

Fig. 13. Visualization of Binary Classification (Table 2) in percent.

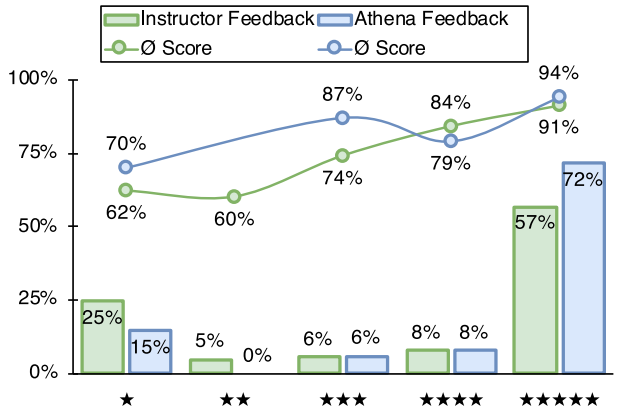

Fig. 14. All ratings for ISE homework exercises by star rating. In this figure, ratings are grouped by *Instructor Feedback* ($n = 428$) or *Athena Feedback* ($n = 102$). The average score in percent is depicted per rating and assessment type. In the study, 530 out of 15, 868 reviews were rated by students.

conclude that Athena does not generate extra efforts through manual interventions required from instructors.

Finding 5 (Specificity): *Athena does not generate unneeded feedback* ($\overline{TNR} = 97\%$).

The specificity of 97% further confirms this, as Athena is good at identifying segments that do not need feedback. Therefore, there is no extra work in removing unnecessary or incorrect feedback.

Finding 6 (Accuracy): *Up to* 78% *of the segments were correctly graded by Athena* ($\overline{Accuracy} = 60\%$).

The accuracy of 60% also supports Finding 3. Future work is needed to improve Athena's coverage, both for the portion of submissions and segments covered in each solution. At the current stage, Athena can help support instructors by contributing partial assessments. However, more work is needed to fulfill the vision of autonomous grading and reach feasible efforts when offering continuous feedback.

8.4. Results: Quality of Athena Feedback compared to instructor feedback

We measured the feedback quality in two ways. First, we asked students to rate their feedback on a 5-star scale. Out of 15,868 total reviews done by the instructors, the students rated 530 reviews. Artemis asks students, "*How useful is the feedback for you?*" displayed underneath their feedback and presents the 5-star scale input. Fig. 14 depicts the distribution by star rating. In the study, 82% of the ratings were either 1-star or 5-star. Students with computer-aided feedback were more likely to give a 5-star rating (72%) when compared to students who received manual feedback (57%). On the same page, computer-aided feedback received 1-star ratings less often (15%) than manual feedback (25%). On average, students giving a 5-star rating (94% and 91%, respectively) had better scores than students giving 1-star ratings (70% and 62%, respectively).

Finding 7 (Perceived Quality): *The computer-aided feedback in Athena has at least the same quality as manual feedback.*

The second measure of feedback quality is students' complaints – or the absence or complaints. Students can complain about their feedback, either requesting a re-evaluation of their solution from a second instructor or requesting more detailed feedback from the same instructor. As re-evaluations are time-intense, students are limited to three complaints in the course; however, legitimate complaints are not counted against this limit. This policy reduces minor or unjustified complaints as submitting a complaint is deemed expensive.

Table 3 outlines the number of submissions for all exercises with Athena feedback and the percentage of complaints. We tested the hypothesis that gradings created using feedback suggestions from Athena lead to fewer complaints than instructor feedback. A Welch Two Sample *t*-test is not suited because the measurements are not normally distributed. Therefore, we employ the Brunner-Munzel Test (Brunner & Munzel, 2000; Neubert & Brunner, 2007), a non parametric statistical test for stochastic equality of two samples. The Brunner-Munzel Test is a generalization of the Mann–Whitney *U* test (Mann & Whitney, 1947; Wilcoxon, 1945) and is suggested as a modern replacement for

Table 3
Student complaints on ISE 2019–2021 distinguishing Athena feedback and instructor feedback. Athena feedback produces significantly fewer complaints than instructor feedback.

Exercise		2019		2020		2021	
		# Sub.	% Compl.	# Sub.	% Compl.	# Sub.	% Compl.
H.11		**1036**		**1125**		**1277**	
	Instructor	1036	0.00%	930	0.86%	1277	1.17%
	Athena	/	/	195	0.00%	/	/
H.12		**943**		**1032**		**1122**	
	Instructor	943	0.42%	744	1.88%	1122	2.76%
	Athena	/	/	288	1.74%	/	/
H.14		**998**		**1103**		**1228**	
	Instructor	998	1.40%	877	2.96%	1228	2.36%
	Athena	/	/	226	2.21%	/	/
H.17		**890**		**1013**		**1112**	
	Instructor	890	1.01%	881	1.70%	1112	2.79%
	Athena	/	/	132	0.76%	/	/
H.18		**943**		**1027**		**1165**	
	Instructor	943	0.32%	662	1.06%	927	3.13%
	Athena	/	/	365	0.55%	238	0.00%
H.19		**950**		**1060**		**1164**	
	Instructor	950	0.53%	1060	5.85%	1164	1.63%
H.20		**832**		**933**		**1068**	
	Instructor	832	1.68%	753	2.39%	881	2.95%
	Athena	/	/	180	1.11%	187	0.00%
H.21		**910**		**1006**		**1176**	
	Instructor	910	3.63%	677	7.68%	1176	8.08%
	Athena	/	/	329	3.65%	/	/
H.22		**877**		**959**		/	
	Instructor	877	1.94%	624	2.72%	/	/
	Athena	/	/	335	3.58%	/	/
H.23		/		/		**1126**	
	Instructor	/	/	/	/	945	3.17%
	Athena	/	/	/	/	181	0.00%
H.24		**898**		**1029**		**1151**	
	Instructor	898	2.23%	700	1.57%	823	4.01%
	Athena	/	/	329	1.82%	328	0.00%
H.25		**882**		**1013**		**1118**	
	Instructor	882	3.74%	767	2.22%	872	4.47%
	Athena	/	/	246	1.63%	246	0.00%

nonparametric tests (Karch, 2021). We use the brunnermunzel R package[7] to compute the test. The Brunner-Munzel Test, based on the complaint rates, results in a test statistic value of $t = 3.8146$. The $p - value$ of the test is 0.000466, which is less than the significance level $\alpha = 0.01$. We can conclude that the Athena-feedback's complaint rate is significantly lower than the Instructor-feedback's complaint rate.

Finding 8 (Quality): ***Feedback generated from Athena leads to fewer student complaints.***

Third, we compare the ratio between segments with and without feedback. We inspect all exercises from 2019 to 2021 and separate the measurements between Athena feedback and instructor feedback.

Finding 9 (Feedback Quantity): ***No evidence suggests that Athena leads to more feedback.***

8.5. Limitations

This section discusses threats to the results' trustworthiness and whether the results are biased based on the researchers' subjective point of view. We distinguish between three aspects of validity: internal validity, external validity, and construct validity (Runeson et al., 2012).

[7] brunnermunzel R package: https://github.com/toshi-ara/brunnermunzel.

8.5.1. Internal validity

The accuracy of the feedback suggestions is measured by the acceptance of the instructor. A second review from a control instructor would allow for a more accurate measurement of accuracy. The instructor might be biased toward confirming a feedback suggestion, requiring less effort than providing a new comment. We noticed that most instructors took the review of the automatic feedback suggestions seriously, but we cannot guarantee that some of the 68 involved teaching assistants failed to review the automatic feedback suggestions thoroughly.

Two authors of this article have been involved in teaching the course ISE and might have influenced the empirical evaluation. However, we tried to separate the research and instructor perspectives. Further, two additional instructors have been involved in the course ISE who are not authors of this paper, and the third author reviewed the results carefully without being involved in the course. In addition, we observed similar results in a second course, which was taught by an independent instructor who was not involved in the research (Bernius et al., 2021).

8.5.2. External validity

Most analyzed exercises have been in the domain of software engineering and computer science in the same university. While we believe that the approach is generalizable for other domains, we have not shown this in this study.

8.5.3. Construct validity

The validity of the ratings might be affected by the question's wording and the score that the students received. Students with a higher score are typically more satisfied and less likely to complain about the quality of the feedback. Therefore, a good rating does not necessarily mean that the feedback was of good quality. Another limitation could be that students like the approach of getting feedback. The ratings measure the perceived quality, which is subjective. We can only infer the quality based on the ratings. Therefore, we consider Finding 7 on the quality of the ratings as anecdotal evidence.

8.6. Discussion

The suggestion coverage of Athena is higher for exercises that do not ask students to come up with their own examples but rather require students to work based on a given problem context. In the exam exercises *E.01*, *E.03*, *E.09*, *E.15 - E.22*, students were asked to extract requirements or use cases from a problem statement. In those exercises, the coverage was mostly above the average, ranging from 48% to 75%. These questions still require students to apply problem-solving skills but limit the variability of the answers. This leads to more similar answers and more reusable feedback.

Exercises asking for examples, such as the ISE homework exercises, have lower Athena suggestion coverage between 13% and 38%. This may be due to the increased variability of answers where students develop their own examples. As Athena tries to find similar text segments, it is more difficult to find a group with shared segments as students choose examples from different problem contexts. Therefore, students are less likely to produce similar answers, and Athena cannot learn to reuse feedback among students.

Athena reuses reviews from instructors. Therefore, the quality of the feedback suggestions depends on the manual feedback provided during the instructor reviews. If instructors provide incorrect manual feedback, Athena will not be able to provide correct feedback suggestions. In the example of ISE, the instructors who review the submission consist primarily of teaching assistants who have limited experience in grading or providing feedback.

Nevertheless, the approach can improve the review process as it allows instructors to handle larger amounts of reviews or to inspect examples. Other systems presented in Section 5 suggest comparing answers only with a sample solution provided by an instructor (Pérez

et al., 2005), thus reducing the variability in the solution space, which might limit the students' creativity. However, creativity is an important aspect of software engineering education (Krusche, Bruegge, et al., 2017).

The use of Athena reduces the workload for instructors and, thereby, enables instructors to better support students individually. All students receive personal attention in the form of Virtual One-To-One (Bernius & Bruegge, 2019) feedback. The efficiency gain in the frequent solution cases results in more time to address specific solutions and take care of problems. Individual feedback is better than presenting a sample solution in a lecture format (Higgins et al., 2002), especially in software engineering, where many creative solutions can co-exist. Individual discussions for different solutions are needed so students and instructors can learn about the benefits, consequences, and trade-offs of new solutions.

9. Conclusion

We have presented an approach that reduces assessment efforts of textual exercises for instructors while scaling feedback for large courses.

The main contributions are: First, a formalization of the assessment effort for a large-scale course using the interactive learning teaching method.

Second, the machine learning-based **framework** "CoFee" outlines how to capture assessment knowledge and automatically suggest feedback. The framework employs segment-based grading and reuses feedback based on segment similarity. We confirmed the frameworks' validity in a laboratory experiment and found that CoFee can reduce the instructors grading effort by 85%.

Third, the **reference implementation** "Athena" demonstrates how to *design* and build *a system that automatically assesses textual exercises* (Artifact Design Goal). Athena uses the ELMo language model to capture core ideas of segments and HDBSCAN clustering to identify groups of similar segments. Athena is open-source software published under the MIT license and integrated into the Artemis system.

Fourth, the **implementation evaluation** describes the usage of Athena in a large-scale software engineering course with up to 2,200 students and up to 68 instructors. The evaluation analyzed the generated feedback and compared feedback given to the students with and without Athena. The findings suggest that Athena can provide feedback for up to 75% of student answers. The feedback suggested is 92% precise and 60% accurate. Athena does not lead to more feedback; however, students perceive the feedback quality as identical, and fewer students complain about Athena grading than manual grading. The evaluation further shows that the accuracy of Athena feedback depends on the type of textual exercise and the variability of possible answers. A higher variance within correct solutions leads to less coverage because of fewer similarities in the student answers.

The article outlines how segment-based structured grading in CoFee allows for collecting and reusing knowledge generated during the manual assessment. Machine learning can support instructors with their assessment work. Working with automated feedback suggestions reduces the assessment efforts and helps instructors deliver consistent feedback and reduce student complaints. Athena does not require training data before grading to learn correct answers and feedback suggestions. Instead, it collects knowledge during the assessment. This incremental process allows instructors to change or introduce new exercises as needed, preventing students from submitting solutions from previous years.

10. Future work

Training based on assessments of past exercises allows Athena to profit from additional knowledge captured in these reviews. However, future work needs to evaluate whether training data from the same exercise in previous years can improve the coverage or accuracy of feedback suggestions.

In addition, the presented research can be extended in four ways: First, additional intermediate representations of text segments can be explored. Athena uses ELMo to capture core ideas within text segments. Further research is needed to explore the accuracy of other types of models, e.g., transformers such as the Bidirectional Encoder Representations from Transformers (BERT) model (Devlin et al., 2019) or the Sparsely Gated Mixture-of-Expert (Jacobs et al., 1991; Lepikhin et al., 2021) based Facebook WMT model (Tran et al., 2021).

Second, by migrating away from a language-dependent language model, CoFee can improve on the current limitation to English answers. Transformer-based models could enable language-independent grading by converting a segment to a language-independent intermediate representation, employing techniques currently used for machine translations. Following this approach would allow CoFee to create a language-independent assessment knowledge and associate feedback to answers independent of the used language. Language-independent grading can allow international students to answer in their preferred language. Instructors can thereby assess work in a foreign language they do not speak themselves.

Third, language models can be fine-tuned by incorporating domain-specific contexts from course materials, such as textbooks, slides, or lecture notes. Customized language models allow CoFee to improve the assessment of exercises requiring a special problem domain knowledge. Transfer Learning could be applied to fine-tune a general-purpose neural network for this specific task (Dai & Le, 2015; Howard & Ruder, 2018). Mayfield and Black (2020) suggest that the relevant world knowledge is already present in pre-trained BERT models.

Forth, another possibility is to combine CoFee's content-based grading with language grading as available in essay scoring systems (cf. Subsection 5.1). The resulting system considers other aspects of the work (e.g., grammar, writing style, and language use) and could extend CoFee's applicability beyond short-answer exercises.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This article is based on previously published papers (Bernius & Bruegge, 2019; Bernius et al., 2020, 2021).

The authors would like to thank Gregor Ziegltrum, Anna Kovaleva, Ngoc-Minh Tran, Clemens Zuck, Adem Khachnaoui, Can Arisan, Jonas Petry, Birtan Gültekin, Linus Michel, Michal Kawka, Maisa Ben Salah, Ndricim Rrapi, Argert Boja, Valerie Bucher, and Tim Cremer for their contributions to Athena as part of their Bachelor or Master theses.

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

Alario-Hoyos, C., Kloos, C., Estévez-Ayres, I., Fernández-Panadero, C., Blasco, J., Pastrana, S., & Villena-Román, J. (2016). Interactive activities: The key to learning programming with MOOCs. In *European stakeholder summit on experiences and best practices in and around MOOCs* (pp. 319–328).

Anderson, L. W., Krathwohl, D. R., Airasian, P. W., Cruikshank, K. A., Mayer, R. E., Pintrich, P. R., Raths, J., & Wittrock, M. C. (2001). *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*. Longmans Green.

Basu, S., Jacobs, C., & Vanderwende, L. (2013). Powergrading: A clustering approach to amplify human effort for short answer grading. *Transactions of the Association for Computational Linguistics, 1*, 391–402. https://doi.org/10.1162/tacl_a_00236

Bernius, J. P., & Bruegge, B. (2019). Toward the automatic assessment of text exercises. In *2nd workshop on innovative software engineering education ISEE '19*. URL: http://ceur-ws.org/Vol-2308/isee2019paper04.pdf.

Bernius, J. P., Kovaleva, A., Krusche, S., & Bruegge, B. (2020). Towards the automation of grading textual student submissions to open-ended questions. In *4th European Conference of software engineering education ECSEE '20* https://doi.org/10.1145/3396802.3396805

Bernius, J. P., Krusche, S., & Bruegge, B. (2021). A machine learning approach for suggesting feedback in textual exercises in large courses. In *8th ACM Conference on learning @ scale L@S '21* https://doi.org/10.1145/3430895.3460135

Biggs, J. (2003). Aligning teaching and assessing to course objectives. *Teaching and learning in higher education: New Trends and Innovations, 2*, 13–17.

Bruegge, B., & Dutoit, A. H. (2009). *Object oriented software engineering using UML, patterns, and java*. Prentice Hall.

Brunner, E., & Munzel, U. (2000). The nonparametric behrens-Fisher problem: Asymptotic theory and a small-sample approximation. *Biometrical Journal, 42*, 17–25. https://doi.org/10.1002/(sici)1521-4036(200001)42:1<17::aid-bimj17>3.0.co;2-u

Chen, X., Breslow, L., & DeBoer, J. (2018). Analyzing productive learning behaviors for students using immediate corrective feedback in a blended learning environment. *Computers & Education, 117*, 59–74. https://doi.org/10.1016/j.compedu.2017.09.013

Cheng, G. (2017). The impact of online automated feedback on students' reflective journal writing in an efl course. *The Internet and Higher Education, 34*, 18–27. https://doi.org/10.1016/j.iheduc.2017.04.002

Dai, A. M., & Le, Q. V. (2015). Semi-supervised sequence learning. In *advances in neural information processing systems, 28*Curran Associates, Inc.. URL: https://proceedings.neurips.cc/paper/2015/file/7137debd45ae4d0ab9aa953017286b20-Paper.pdf.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *2019 conference of the North American chapter of the association for computational linguistics: Human language technologies, 1* pp. 4171–4186). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-1423 *(Long and Short Papers)*.

Feynman, R. P. (1994). *Six easy pieces*. Basic Books.

Förster, M., Weiser, C., & Maur, A. (2018). How feedback provided by voluntary electronic quizzes affects learning outcomes of university students in large classes. *Computers & Education, 121*, 100–114. https://doi.org/10.1016/j.compedu.2018.02.012

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design patterns: Elements of reusable object-oriented software*. Addison-Wesley.

Higgins, R., Hartley, P., & Skelton, A. (2002). The conscientious consumer: Reconsidering the role of assessment feedback in student learning. *Studies in Higher Education, 27*, 53–64. https://doi.org/10.1080/03075070120099368

Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification. In *56th annual meeting of the association for computational linguistics, 1* pp. 328–339). Association for Computational Linguistics volume. https://doi.org/10.18653/v1/P18-1031. Long Papers.

Irons, A. (2007). *Enhancing learning through formative assessment and feedback*. Routledge.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation, 3*, 79–87. https://doi.org/10.1162/neco.1991.3.1.79

Jensen, L. X., Bearman, M., & Boud, D. (2021). Understanding feedback in online learning – a critical review and metaphor analysis. *Computers & Education, 173*, Article 104271. https://doi.org/10.1016/j.compedu.2021.104271

Johanßen, J. O. (2019). *Continuous user Understanding in software evolution*. Dissertation Technische Universität München München. URL: http://d-nb.info/1201482682/34.

Karch, J. D. (2021). Psychologists should use brunner-munzel's instead of mann-whitney's u test as the default nonparametric procedure. *Advances in Methods and Practices in Psychological Science, 4*, Article 251524592199960. https://doi.org/10.1177/2515245921999602

Kirschner, P., Sweller, J., & Clark, R. (2006). Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching. *Educational Psychologist, 41*, 75–86. https://doi.org/10.1207/s15326985ep4102_1

Krusche, S., Bruegge, B., Camilleri, I., Krinkin, K., Seitz, A., & Wöbker, C. (2017). Chaordic learning: A case study. In *39th international Conference on software engineering: Software engineering Education and training ICSE-SEET '17* (pp. 87–96). IEEE. https://doi.org/10.1109/ICSE-SEET.2017.21.

Krusche, S., & Seitz, A. (2018). ArTEMiS: An automatic assessment management system for interactive learning. In *49th ACM technical symposium on computer science education (SIGCSE)* (pp. 284–289).

Krusche, S., & Seitz, A. (2019). Increasing the interactivity in software engineering moocs - a case study. In *52nd Hawaii international conference on system sciences* (pp. 1–10).

Krusche, S., Seitz, A., Börstler, J., & Bruegge, B. (2017). Interactive learning: Increasing student participation through shorter exercise cycles. In *19th Australasian computing education conference* (pp. 17–26). ACM.

Krusche, S., von Frankenberg, N., & Afifi, S. (2017). Experiences of a software engineering course based on interactive learning. In *Tagungsband des 15. Workshops Software Engineering im Unterricht der Hochschulen (SEUH)* (pp. 32–40). CEUR.

Krusche, S., von Frankenberg, N., Reimer, L. M., & Bruegge, B. (2020). An interactive learning method to engage students in modeling. In *International conference on software engineering: Software engineering education and training* (pp. 12–22).

Lepikhin, D., Lee, H., Xu, Y., Chen, D., Firat, O., Huang, Y., Krikun, M., Shazeer, N., & Chen, Z. (2021). GShard: Scaling giant models with conditional computation and automatic sharding. In *International conference on learning representations*. URL: https://openreview.net/forum?id=qrwe7XHTmYb.

Li, Y., & Yang, T. (2018). Word embedding for understanding natural language: A survey. In S. Srinivasan (Ed.), *Guide to big data applications* (pp. 83–104). Cham: Springer. https://doi.org/10.1007/978-3-319-53817-4_4.

Mann, H. B., & Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics, 18*, 50–60. https://doi.org/10.1214/aoms/1177730491

Mayfield, E., & Black, A. W. (2020). Should you fine-tune BERT for automated essay scoring?. In *15th workshop on innovative use of NLP for building educational applications* (pp. 151–162). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.bea-1.15.

McInnes, L., & Healy, J. (2017). Accelerated hierarchical density based clustering. In *International conference on data mining workshops* (pp. 33–42). https://doi.org/10.1109/ICDMW.2017.12

Mitchell, T., Russell, T., Broomhead, P., & Aldridge, N. (2002). Towards robust computerised marking of free-text responses. In *6th international computer assisted assessment (CAA) conference*. UK: Loughborough University.

Neubert, K., & Brunner, E. (2007). A studentized permutation test for the non-parametric behrens-Fisher problem. *Computational Statistics & Data Analysis, 51*, 5192–5204. https://doi.org/10.1016/j.csda.2006.05.024

Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global vectors for word representation. In *Conference on empirical methods in natural language processing* (pp. 1532–1543). Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1162.

Pérez, D., Gliozzo, A. M., Strapparava, C., Alfonseca, E., Rodríguez, P., & Magnini, B. (2005). Automatic assessment of students' free-text answers underpinned by the combination of a bleu-inspired algorithm and latent semantic analysis. In *18th international Florida Artificial intelligence research society conference* (pp. 358–363). AAAI Press. URL: http://www.aaai.org/Library/FLAIRS/2005/flairs05-059.php.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. In *Conference of the North American chapter of the association for computational linguistics: Human language technologies* (pp. 2227–2237). Association for Computational Linguistics. https://doi.org/10.18653/v1/N18-1202.

Popper, K. R. (1934). *Logik der Forschung – Zur Erkenntnistheorie der modernen Naturwissenschaft*. Springer.

Popper, K. R. (1959). *The logic of scientific discovery*. *Hutchinson*.

Pulman, S. G., & Sukkarieh, J. Z. (2005). Automatic short answer marking. In *2nd Workshop on building educational applications using NLP EdAppsNLP 05* (pp. 9–16). Association for Computational Linguistics. https://doi.org/10.5555/1609829.1609831.

Ramos, J. (2003). Using TF-IDF to determine word relevance in document queries. In *1st instructional conference on machine learning, 242* pp. 1–4).

Runeson, P., Höst, M., Rainer, A., & Regnell, B. (2012). *Case study research in software engineering*. John Wiley & Sons, Inc. https://doi.org/10.1002/9781118181034

Singh, A., Karayev, S., Gutowski, K., & Abbeel, P. (2017). Gradescope: A fast, flexible, and fair system for scalable assessment of handwritten work. In *4th Conference on learning @ scale L@S '17* (pp. 81–88). ACM. https://doi.org/10.1145/3051457.3051466.

Sukkarieh, J., Pulman, S. G., & Raikes, N. (2003). Auto-marking: Using computational linguistics to score short, free-text responses. In *29th Annual Conference of the international Association for educational assessment IAEA* (pp. 1–15).

Sweller, J., & Cooper, G. A. (1985). The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and Instruction, 2*, 59–89. https://doi.org/10.1207/s1532690xci0201_3

Trafton, J. G., & Reiser, B. J. (1993). *Studying examples and solving problems: Contributions to skill acquisition*. Washington, DC, USA: Technical Report Naval HCI Research Lab.

Tran, C., Bhosale, S., Cross, J., Koehn, P., Edunov, S., & Fan, A. (2021). Facebook AI's WMT21 news translation task submission. In *6th conference on machine translation* (pp. 205–215). Association for Computational Linguistics. URL: https://aclanthology.org/2021.wmt-1.19.

VanLehn, K. (1996). Cognitive skill acquisition. *Annual Review of Psychology, 47*, 513–539. https://doi.org/10.1146/annurev.psych.47.1.513

Wieringa, R. J. (2014). *Design science methodology for information systems and software engineering*. Springer. https://doi.org/10.1007/978-3-662-43839-8

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometric Bulletin, 1*, 80–83. https://doi.org/10.2307/3001968

Williams, R., & Haladyna, T. (1982). Logical operations for generating intended questions (logiq): A typology for higher level test items. In G. H. Roid, & T. M. Haladyna (Eds.), *Toward a technology of test-item writing* (pp. 161–187). New York: Academic Press.

Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data mining: Practical machine learning tools and techniques*. Elsevier.