TUM School of Computation, Information and Technology
Technical University of Munich

Dissertation

Automatic Assessment of Textual Exercises

Jan Philip Bernius

TUM School of Computation, Information and Technology
Technische Universität München

Automatic Assessment of Textual Exercises

Jan Philip Bernius

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Die Dissertation wurde am 27.06.2022 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 31.10.2022 angenommen.

"I think, however, that there isn't any solution to this problem of education other than to realize that the best teaching can be done only when there is a direct individual relationship between a student and a good teacher – a situation in which the student discusses the ideas, think about the things, and talk about the things. It's impossible to learn very much by simply sitting in a lecture, or even by simply doing problems that are assigned. ***But in our modern times we have so many students to teach that we have to try to find some substitute for the ideal.****"*

— Richard P. Feynman [Fey94]

Acknowledgments

First and foremost, I would like to express my deepest appreciation to Prof. Dr. Bernd Brügge for advising me for the past four years. Bernd has inspired me since early in my bachelor's studies. When we first discussed a potential doctoral thesis, he told me there is only a 1% chance of making this work. Thank you for turning my 1% chance into 100%. I'm extremely grateful to Prof. Dr. Bastian Tenbergen for serving as the second advisor of my doctoral studies. This endeavor would not have been possible without my mentor Prof. Dr. Stephan Krusche. I want to thank Stephan for the collaboration on Artemis, our large-scale teaching of "Introduction to Software Engineering" and "Patterns in Software Engineering," and his feedback on all aspects of my research project.

Many thanks to Dr. Dora Dzvonyar, Dr. Jan Ole Johanßen, Dr. Mariana Avezum, Dr. Dominic Henze, Dr. Paul Schmiedmayer, Dr. Sebastian Klepper, Dr. Nadine von Frankenberg und Ludwigsdorff, Marko Jovanović, Jonas Schulte-Coerne, Dr. Lara Marie Reimer, Jens Klinker, Maximilian Kapsecker, and Evgeny Volynsky for being part of my journey as a doctoral student and forming my time at the Chair for Applied Software Engineering. I want to especially thank Helma Schneider, Uta Weber, Monika Markl, and Ruth Demmel for keeping the chair running and pulling every string necessary to make all crazy ideas possible. I am grateful to Matthias Linhuber, Florian Angermeir, Vincent Picking, and Robert Jandow for maintaining the IT infrastructure to run Athena.

I had the pleasure of working with GregorZiegltrum, 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 on Athena as part of their Bachelor's or Master's theses. I would like to recognize all tutors of the "Introduction to Software Engineering" course in 2020 and 2021 for assessing textual exercises with Athena and all students who submitted their solutions. I am also grateful to Prof. Dr. Uwe Baumgarten and David Werner for volunteering to use and allowing me to evaluate Athena as part of their exam assessment in "Networks for Monetary Transactions."

I am extremely grateful to Dr. Lara Marie Reimer for the last seven years and for having a partner I can discuss work and life related topics with. I am forever grateful to my parents for laying the foundation for my academic career. I would not have gotten this far without their support.

Abstract

Engineering disciplines require problem-solving skills. Open-ended textual exercises allow students to acquire these skills. Memorization alone is not sufficient to master problem solving skills. Problem solving requires interactivity. Students can learn from their mistakes when instructors provide individual feedback. However, interactive learning leads to more grading, which is a manual, repetitive, and time-consuming activity. In addition, the number of engineering students graduating per year has steadily increased over the past decade. This increase has resulted in large courses, which further increases the assessment workload for instructors.

This dissertation presents four publications which introduce the Athena system to reduce the grading workload by automatically assessing textual exercise submissions: Publication [BB19] presents the overall approach to collect assessment knowledge from manual segment-based assessments. Publication [Ber+20] introduces an algorithm to decompose student answers into topically-coherent segments. Publication [BKB21] describes the architecture of the Athena system, which automates grading based on topic modeling, language models, and an assessment knowledge repository. Publication [BKB22] evaluates the Athena system in a large course with 34 textual exercises offered between 2019 and 2021 for up to 2,200 enrolled students. More than 47,500 interactive open-ended textual exercise submissions were processed by Athena during this period. Athena suggested automatic feedback for 45% of these submissions. Instructors accepted 92% of Athena's suggestions.

Zusammenfassung

Ingenieurwissenschaften erfordern Problemlösungskompetenz. Freitextaufgaben ermöglichen es den Studierenden, diese Fähigkeiten zu erwerben. Auswendiglernen allein reicht nicht aus, um Problemlösungskompetenz zu erlernen. Das Lösen von Problemen erfordert Interaktivität. Studierende können aus ihren Fehlern lernen, wenn die Lehrenden individuelles Feedback geben. Interaktives Lernen führt jedoch zu mehr Korrekturen, die eine manuelle, sich wiederholende und zeitaufwändige Tätigkeit sind. Hinzu kommt, dass die Zahl der jährlichen Absolventen von Ingenieurstudiengängen in den letzten zehn Jahren stetig gestiegen ist. Dieser Anstieg hat zu großen Kursen geführt, was die Arbeitsbelastung der Lehrenden durch Korrekturen weiter erhöht.

Diese Dissertation stellt in vier Publikationen das System Athena vor, das den Korrekturaufwand durch automatische Bewertung der eingereichten Textaufgaben reduziert: Publikation [BB19] stellt den Gesamtansatz zur Sammlung von Bewertungswissen aus manuellen segmentbasierten Bewertungen vor. Die Publikation [Ber+20] stellt einen Algorithmus zur Zerlegung von Studierendenantworten in thematisch kohärente Segmente vor. Publikation [BKB21] beschreibt die Architektur des Athena Systems, das die Korrektur basierend auf Topic Modeling, Sprachmodellen und bestehenden Korrekturen automatisiert. Publikation [BKB22] evaluiert das Athena System in einem großen Kurs mit 34 Textübungen, der zwischen 2019 und 2021 für bis zu 2.200 eingeschriebene Studierende angeboten wurde. Mehr als 47.500 interaktive, offene Textübungen wurden in diesem Zeitraum mit Athena bearbeitet. Athena schlug für 45 % der Abgaben Feedback vor. Die Lehrenden akzeptierten 92 % der Vorschläge von Athena.

Publication Preface

The contribution of this dissertation is based on the following three peer-reviewed first-author publications in international conferences and journals:

Publication [Ber+20]
Jan Philip Bernius, Anna Kovaleva, Stephan Krusche, and Bernd Bruegge. "Towards the Automation of Grading Textual Student Submissions to Open-ended Questions." In: *4th European Conference of Software Engineering Education*. ECSEE '20. Seeon, Germany: Association for Computing Machinery (ACM), June 2020, pp. 61–70. ISBN: 9781450377522. DOI: `10.1145/3396802.3396805`

Publication [BKB21]
Jan Philip Bernius, Stephan Krusche, and Bernd Bruegge. "A Machine Learning Approach for Suggesting Feedback in Textual Exercises in Large Courses." In: *8th ACM Conference on Learning @ Scale*. L@S '21. Potsdam, Germany: Association for Computing Machinery (ACM), June 2021, pp. 173–182. ISBN: 9781450382151. DOI: `10.1145/3430895.3460135`

Publication [BKB22]
Jan Philip Bernius, Stephan Krusche, and Bernd Bruegge. "Machine Learning Based Feedback on Textual Student Answers in Large Courses." In: *Computers and Education: Artificial Intelligence* 3 (June 2022). ISSN: 2666-920X. DOI: `10.1016/j.caeai.2022.100081`

The following first-author international workshop paper is included to provide additional context:

Publication [BB19]
Jan Philip Bernius and Bernd Bruegge. "Toward the Automatic Assessment of Text Exercises." In: *2nd Workshop on Innovative Software Engineering Education*. ISEE '19. Stuttgart, Germany: CEUR-WS.org, February 2019, pp. 19–22. URL: `http://ceur-ws.org/Vol-2308/isee2019paper04.pdf`

The following peer-reviewed first-author publications are related to the dissertation, but not embedded into the dissertation:

Publication [BKB20]
Jan Philip Bernius, Anna Kovaleva, and Bernd Bruegge. "Segmenting Student Answers to Textual Exercises Based on Topic Modeling." In: *17th Workshop on Software Engineering im Unterricht der Hochschulen.* SEUH '20. Innsbruck, Austria: CEUR-WS.org, February 2020, pp. 72–73. URL: `http://ceur-ws.org/Vol-2531/poster03.pdf`

Publication [Ber21]
Jan Philip Bernius. "Toward Computer-Aided Assessment of Textual Exercises in Very Large Courses." In: *52nd ACM Technical Symposium on Computer Science Education.* SIGCSE '21. Toronto, ON, Canada: Association for Computing Machinery (ACM), March 2021, p. 1386. DOI: `10.1145/3408877.3439703`

Contents

1 Introduction

Computer science and software engineering are problem-solving activities that require creativity, collaboration, and practical application of knowledge [AK97]. However, creative activities such as modeling are hard to teach and assess, particularly in very large courses with thousands of students. The number of enrolled students in computer science courses has increased significantly, some by a factor of five to ten in the last 20 years. Another problem is to assess how the students apply problem-solving activities in an exam. Much expertise is necessary to assess whether the solution is complete, unambiguous, and correct and whether it addresses the problem to be solved. Handing out a sample solution and providing generic feedback does not help because there is often not a single correct solution. Moreover, even if there is only one solution, the iterative aspect of moving toward that solution requires individual feedback from the instructor. The job of a good instructor is also to identify creative solutions that do not follow the sample solution.

For very large classes, instructors also face an assessment problem. An easy way out is multiple-choice questions because they are easy to grade, and the grading process can easily be automated. However, multiple-choice questions are not the answer for assessing questions that require creative thinking and often lead to individual solutions. Further, trivial exercises do not stimulate higher cognitive skills and do not reflect an engineer's daily work [KS19]. The challenge is to create textual questions to stimulate higher cognitive skills, such as problem-solving, which are important in computer science. Software engineering students need to learn constructive and creative capabilities. Grading these types of questions causes a high assessment effort on the instructor's side.

Learning methodologies such as active learning, blended learning, experiential learning, or flipped classrooms have been created to address these problems, but their applicability to very large classrooms is limited. This dissertation presents Computer-aided Feedback for textual exercises (CoFee), a framework to generate and suggest automated 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.

The dissertation is publication-based and structured as follows: Chapter 1 provides an overview of textual exercises (1.1), assessment (1.2), and the evolution of teaching (1.3). Chapter 2 introduces Natural Language Processing with its history (2.1), topic segmentation and topic modeling (2.2), and language models (2.3). Chapter 3 introduces clustering algorithms. Chapter 4 outlines the research methodology used in this dissertation (4.1) and explains the courses used for empirical evaluation (4.2). Chapter 5 contains the workshop paper "Toward the Automatic Assessment of Text Exercises" and outlines a machine learning-based approach to assess textual exercises automatically. Chapter 6 contains the conference paper "Towards the Automation of Grading Textual Student Submissions to Open-ended Questions," which presents the design and implementation of an algorithm using topic modeling for segmenting solutions into short segments for assessment. Chapter 7 contains the conference paper "A Machine Learning Approach for Suggesting Feedback in Textual Exercises in Large Courses," which introduces the reference implementation "Athena" and presents an empirical evaluation based on data from two courses from 2020. Chapter 8 contains the journal article "Machine Learning Based Feedback on Textual Student Answers in Large Courses," which formalizes the research process following the design science methodology. Further, the article extends the evaluation based on an extended dataset over three years and with more statistics. Finally, Chapter 9 summarizes the dissertation's contributions and provides an outlook on future work.

1.1 Textual Exercises

The instructor needs to facilitate the problem-solving learning process [DKM98]. Exercises are a proven method to train higher cognitive skills, including the acquisition of domain-specific knowledge, analysis and design methods, and the evaluation of the results.

"Recall" or "open" questions [GL08] focus on the respondent coming up with an answer of their own to a posed question. "Open" questions usually provide benefits to higher levels of Bloom's learning objectives [KA10], namely "Analyze," "Apply," and "Evaluate." Text exercises can be differentiated into three main categories: (1) Structural exercises focus on solving a notation-based math equation or writing a piece of computer code. (2) Natural language exercises require respondents to answer a question using written language. (3) Speech exercises test students based on their comprehension of a spoken passage.

Natural language questions can be separated into fill-the-blank, short-answer questions, definition queries, reading comprehension, or essays. The main dimension of distinguishment is the length of a response. While fill-the-blank answers often only require a few words, which can be automatically verified against a set of options, the difference between short-answer questions and essays is not well defined. Siddiqi, Harrison, and Siddiqi [SHS10] define the length of a short answer to be three to four sentences, while Sukkarieh and Stoyanchev [SS09] define an optimal length to be under 100 words. The length of an essay is usually defined as multiple paragraphs up to several pages [BGS15]. Another criterion that should be considered when differentiating between short answers and essays is the focus of each exercise: Although the content of an essay is not negligible, the main priority of an essay lies in determining the proficiency of an examinee as a writer as well as the linguistic skills demonstrated in answering an open-ended question with no clear, definite right or wrong answer [JM09]. In comparison, short answer questions are graded based on the response's content, and an examinee's writing style and grammatical or spelling errors are omitted. Also, short answer questions require the examinee to answer by stating clear facts and proven factual statements, leaving out their personal opinion and vague statements or assumptions.

Reading comprehension exercises and exercises, where the definition of a concept is obtained, are not considered in this work since they often only address lower levels of Bloom's taxonomy and cannot be classified as "open" questions where the examinee has to come up with an own answer.

1.2 Assessment

Concrete problem-solving strategies are taught in paradigms accepted by the profession [Kuh96]. Each paradigm provides a set of problem-solving exercises. These are usual (textual) exercises that involve applying problem-solving techniques. Exercises help students to learn, understand and apply a paradigm.

Instructors send feedback messages to students about correct and incorrect elements of their solution; students use the information from the feedback message to improve their solution [NM06]. Students' overall satisfaction correlates strongly with the quality and quantity of feedback they receive [JHG13]. Students engage most with frequent, detailed, and timely feedback focused on their task performance [GS05]. Feedback must arrive timely to be relevant for further learning or to be helpful with ongoing and upcoming exercises.

Feedback in the form of a dialogue engages the students the most, as the students can address the feedback instantly, and instructors can engage in a discussion [NM06; Car+10; PHM11]. Furthermore, within a dialog, instructors can guide and encourage students with their feedback and can foster a positive learner identity.

As this work's goal is to provide students with feedback for an exercise solved through written text, defining the type of feedback to provide is also required. Exercise types such as multiple choice questions or programming exercises are typically assessed binary as right or wrong, thus allowing students to recognize their mistakes immediately. The openness of a text exercise demands concrete reasoning as to why a particular grade has been given. We consider written comments by an instructor next to a numeric score to be an obligatory part of the feedback. Comments allow instructors to justify their feedback and create a classroom environment where they explain their reasoning behind an assessment instead of just providing a score.

1.3 Evolution of Teaching

In ancient Greece, from 335 B.C., the peripatetic school teaches philosophy in discussions while walking. The peripatetic school was named after the Peripatos, a pathway encircling the Acropolis in Athens. Aristotle's teaching consisted of two activities: First, conversing with his more mature pupil, allegedly lecturing while walking. Second, giving lectures on philosophy and rhetoric to a larger audience. [Sey82; Sey91]

The low ratio between teachers and students enables them to interact directly during their discussions. The peripatetic school was teaching a static set of knowledge defined in the doctrines laid down by Aristotle and maintained by his followers. Figure 1.1 depicts a teaching session by Aristoteles.

Figure 1.1: The School of Athens, fresco painted between 1509 and 1511 by Raffaello Sanzio da Urbino (public domain).

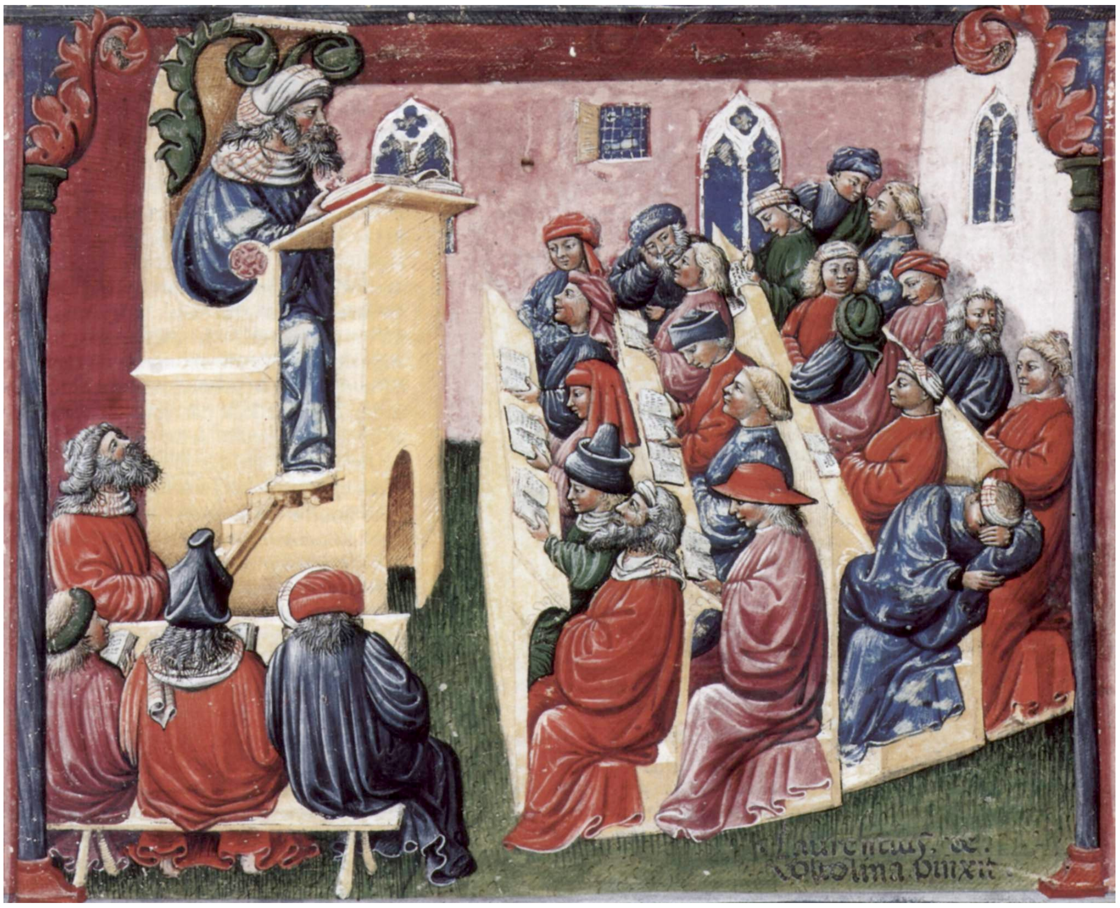


Figure 1.2: Lecture at the University of Bologna in the 1350s. Painting by Laurentius de Voltolina (public domain).

Scholasticism was the medieval school of philosophy and the dominating form of higher education between 1100 and 1700 A.D. in Europe [Mar03]. The methodology originated from catholic monastic schools and is not limited to teaching philosophy or theology. A teaching unit consists of three phases: First, in *lectio*, a teacher reads an authoritative text and follows up with a commentary on the text. No questions are permitted during the lectio. Second, students reflect on the text during *meditatio*. Third, during *quaestio*, students can ask the questions that occurred to them during meditatio. Figure 1.2 depicts a lecture setting at the University of Bologna from the second half of the 14^{th} century.

Today, teaching is primarily conducted through lectures in which a cohort of students follows a recitation of the professor in a big lecture room. On top of that, some courses offer additional sessions with smaller group sizes conducted by teaching assistants, either doctoral students of the professor or students from previous course editions. Richard P. Feynman, a distinguished teacher, and Nobel Prize winner, applied the same teaching paradigm 50 years ago in 1963 at the California Institute of Technology in his fundamental physics lectures.

"*The whole group of 180 students gathered in a big lecture room twice a week to hear these lectures and then they broke up into small groups of 15 to 20 students in recitation sections under the guidance of a teaching assistant*" [Fey94].

Student numbers rise in university courses. An extreme example is the computer science program at the Technical University of Munich (TUM). The first-year course Introduction to Software Engineering (EIST) increased by a factor of 5 within ten years. Figure 1.3 visualizes this growth in student numbers, passing 2,000 in 2021. Traditional university teaching is based on real-time communication and requires students to be present to participate. Class interaction is getting more difficult as more teaching staff is needed to handle the growing student numbers. Further, new communication methods with large audiences are needed [KG17]. Equipping students with clickers to answer questions live during lectures was one of the first interventions [May+09]. OnlineTED[1] is a commercial product used at

[1]OnlineTED: `https://onlineted.de`

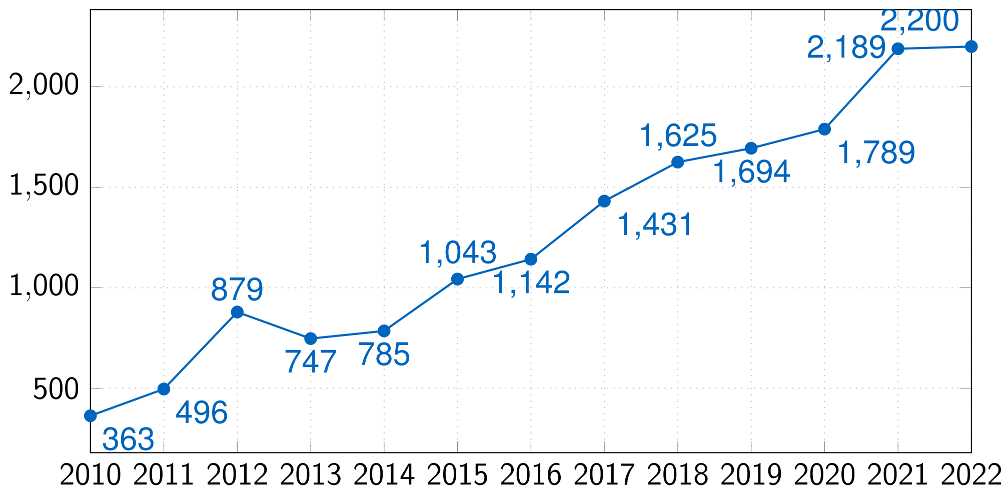


Figure 1.3: The number of registered students in the course EIST between 2010 and 2022.

the TUM School of Management, which allows students to use their Smartphones as a clicker. Clickers are a method to force student interaction by introducing a response system. Students click an answer to a multiple-choice question, and the instructor can evaluate the answers and discuss the rationale behind the correct answer [May+09]. Students that participate in discussions and ask questions during the lecture apply their knowledge constantly and gain a deeper understanding of the course contents [BE91].

Massive Open Online Courses (MOOCs) are open-access online courses that allow an unlimited number of participants. MOOCs require an instructional design with scalable interactions and feedback to handle the massive number of participants. Common techniques used in MOOCs are forums, chat rooms, peer grading, and automated feedback (primarily multiple choice). Unfortunately, the course sizes make it infeasible to provide instructor feedback to all students.

Students are typically easier graders than professional instructors, e.g., awarding 25% more points in peer-graded practice exams than professionals. This grading gap is especially noticeable for higher-level cognitive skills measured in Bloom's taxonomy. Exercises of upper levels, e.g., in "Apply" or "Analyze," are harder to solve and grade for students. [FP10]

Sebastian Thrun and Peter Norvig from Stanford University offered their 2011 course "Introduction to Artificial Intelligence" as a MOOC for free. More than 160, 000 students from 190 countries participate in the online course. The online platform Udacity[2] was a side result of the Stanford course and is now one of the most-known platforms for online courses. Coursera[3] is another online course platform founded by Stanford professors Andrew Ng and Daphne Koller at a similar time as Udacity [Sev12]. The MOOC platform edX[4] by the Massachusetts Institute of Technology and Harvard University provides university-level online learning to 40 million worldwide users [Rod12]. edX is powered by the Open edX Learning Management System (LMS).

[2]Udacity: `https://udacity.com`
[3]Coursera: `www.coursera.org`
[4]edX: `www.edx.org`

2 Natural Language Processing

Natural Language Processing (NLP) is concerned with computers processing and analyzing natural language. NLP is a subfield of computer science, artificial intelligence, and linguistics. The goal of NLP is to understand the contents of documents and the meaning of language in a given context. Typical NLP tasks include knowledge extraction, categorization, question answering, speech recognition, language generation, and translation. NLP emerged in the 1950s. Section 2.1 provides a brief introduction to the history of NLP. The following sections describe the NLP methods used by Athena.

Athena splits answers into meaningful units for granular feedback. For this reason, Section 2.2 describes Topic Segmentation and focuses on methods and approaches to segment text into meaningful units. Athena uses word embeddings from a language model to understand and compare student answers. Therefore, Section 2.3 describes language models, word embeddings, and transformers, namely the instantiations word2vec, ELMo, BERT, and WMT. Language models express a probability distribution of words occurring in the same context. Word embeddings represent the meaning of words in the form of a vector. A transformer is a deep learning model commonly used for NLP and computer vision tasks.

2.1 History

Since the introduction of the Turing Test by Alan Turing [Tur50] in 1950, scientists have been trying to develop an algorithm that could translate, generate, and analyze text in a way identical to a human being's natural speech. Noam Chomsky made the first attempt to formalize natural language leading to the Chomsky hierarchy. The Chomsky hierarchy defines a hierarchy of types of formal grammar [Cho56].

Type-0 (“unrestricted”) grammars include all formal grammar. They generate exactly all languages that a Turing machine can recognize. Type-1 (“context-sensitive”) grammars describe the syntax of natural language where a word may or may not be appropriate in a certain place depending on the context and can be recognized by linear-bounded non-deterministic Turing machines. Type-2 (“context-free”) grammars generate languages that a non-deterministic pushdown automaton can recognize and form the basis for the structure of most programming languages. Type-3 (“regular”) grammars generate the regular languages. Regular expressions defining a search schema for text are a well-known example of a very limited Chomsky type-3 grammar. Chomsky tried to develop a grammar that could generate all kinds of English sentences [Cho56]. Though his work did not result in an all-embracing grammar, his attempts contributed immensely to linguistics and further research on teaching language to computers. The theory he invented is called the “production rule” [CW14].

Roger Schank was among the first to attempt the processing of natural language and introduced the conceptual dependency theory for natural language understanding [ST69]. Conceptual dependency theory outlines a model that represents knowledge independent of the use of words. The model can capture different formulations of sentences representing the same meaning.

Following the formalization of language and representation of knowledge, the next challenge was machine translation [CCG81] of natural language. Jaime Carbonell generalized Schank’s ideas and applied them to areas such as text mining, information retrieval [Car80; Car+97], summarization, [GC98; Gol+99; Mit+99] free-text question-answering [Nyb+02; Nyb+03; LC04a; LC04b], and related tasks.

The first application of efficient backpropagation was described in 1981 by Paul Werbos [Wer81]. Then, David Rumelhart elaborated on and popularized backpropagation [RHW86a; RHW86b] by demonstrating the experimental use of internal representations in hidden layers.

In 2010, the advancements of cheap, powerful GPU-based computing systems allowed to overcome the fundamental deep-learning problem of gradient descent by being able to propagate errors over a few layers within a reasonable time.

This computational work was not feasible in the early 1990s, given the limited computational power of CPUs at the time. [Sch15] Since 2010, neural networks have a widened variance of applications such as speech recognition, machine vision, autonomous driving, and NLP.

2.2 Topic segmentation

This section discusses the literature, mainly focusing on text segmentation, information retrieval, and topic modeling. Text segmentation is one of the NLP tasks concerned with dividing a text into meaningful units. The term changed its meaning over time. Before 2000, text segmentation was primarily concerned with processing scanned documents and segmenting typed or handwritten textual contents from figures or blank spaces [EL90]. The main goal was to identify areas where optical character recognition could be applied [JB92]. After 2000, the term became more commonly used to describe the process of subdividing strings into meaningful units. The field subdivides into word segmentation, intent segmentation, sentence segmentation, and topic segmentation. The remainder of this section focuses on topic segmentation.

Text segmentation can be applied to various NLP tasks, including emotion extraction [Wu+07], sentiment mining [Gao+10], text summarization [LWZ06; CW14], topic identification [BCT02; FWA07], and information retrieval [Hua+03]. [PT17] Information retrieval concerns finding material from an unstructured collection of documents that satisfy an information need [Rij77; MRS08; JR10].

The literature defines different results of text segmentation: A segment is most often defined as a word. Other frequent definitions of segments are characters, topics, sentences, or lines. Less frequently, segments are defined as phrases, paragraphs, or tags. [PT17] This dissertation defines a segment as a topically coherent phrase. A segment can span from a clause to multiple sentences.

Segmentation approaches can be divided into linear approaches and hierarchical approaches [Yaa97; EB08]. Linear approaches produce a linear series of segments, where segments may not overlap each other. Hierarchical approaches provide a hierarchical relationship of segments where segments can be broken down into smaller segments. Linear segmentation is more commonly used in the literature. Hierarchical segmentation can be used in discourse retrieval. [PT17] The approach presented in Chapter 6 [Ber+20] uses a hierarchical structure of segments to identify topically coherent segments by splitting them into clauses or merging sentences. However, the result is a linear segmentation.

2.2.1 Lexical Cohesion

Marti Hearst was among the first to present an unsupervised method for text segmentation with TextTiling [Hea97], which still serves as the baseline many years later [JCN13]. TextTiling derives topic boundaries using tokenization, lexical score determination, and depth score computation. Hearst treats text as a two-level structure that defines several topics and subtopics. TextTiling divides a text between 1800 and 2500 words into segments that belong to different subtopics. [Hea97]

Lexical cohesion, introduced by Michael Halliday, can determine subtopics, as similar vocabulary usually indicates a coherent part of the text [HH76].

However, this approach cannot handle multiple different subtopics co-occurring in the same part of the text. That is why Hearst separated the process into three activities instead of directly calculating lexical chains: (1) tokenization into terms and sentence-sized units, (2) determination of a score for each sentence-sized unit, and (3) detection of the subtopic boundaries [Hea97]. Tokenization focuses on preprocessing the text, such as removing stop words, reducing the words to their lexical root, and converting everything to lowercase. Stop words are words that do not contribute to the semantics of the text, i.e., the verb "to be" or conjunctions. Lexical scores are determined by two different methods: blocks of text and vocabulary introduction. Both methods compare the adjacent topics of the potential subtopic boundary and search for a topic shift rather than the topic itself. The "blocks of text" method compares vectors containing the number of times a lexical item occurs within a block. The vocabulary introduction method counts the number of first-time uses of words in the current token sequence. The depth score identifies local minima in a series of potential topic gaps. A local minima indicates that two adjacent blocks are only marginally related.

2.2.2 Multi-Score Algorithm Based on Cue Phrases

Doug Beeferman introduced a multi-score algorithm that uses cue phrases instead of lexical cohesion [BBL97]. Based on a dataset of 188 million words from Wall Street Journal articles and a collection of news and broadcast transcripts, the authors identified words indicating a topic shift. The algorithm tries to learn the most valuable clues from the text and then calculates a statistical model. Two approaches are combined into a multi-score algorithm called short- and long-range models of language [BBL97].

2.2.3 Combination of Lexical Cohesion and Cue Phrases

Jeffrey Reynar combines lexical cohesion and cue phrases to a new approach [Rey99]. To account for ambiguities in words (i.e., wild plant as in flower vs. chemical plant as in factory), Reynar applies lexical cohesion on bigram-word frequencies in his Word Frequency Algorithm. The algorithm predicts whether the words that follow a topic boundary are generated dependent or independent of the proceeding words.

The second algorithm, the "Maximum Entropy Model" uses seven features to predict topic boundaries. Word frequencies, domain cues, pronouns, named entities, and first uses of words as an indicator for a continuation of a topic. For instance, they check for an occurrence of a pronoun within the first five words of a sentence as an indicator for a coherent block. [Rey99]

2.2.4 Similarity Ranking and Divisive Clustering

Freddy Choi [Cho00] builds on top of Reynar 's algorithm [Rey99] and adjusts the similarity measure and location of topic boundaries. The cosine distance between vectors of word frequency can be disproportionally increased by one additional word occurrence. Especially given that different parts of a text are less coherent, the cosine method is inappropriate to compare segments from different parts of a text. Therefore, Choi proposes to convert the cosine similarity value into a rank of similarity among neighboring segments [Cho00]. Divisive clustering determines the topic boundaries based on the Maximum Entropy Model [Rey99]. This algorithm is two times more accurate and seven times faster than Hearst 's sliding window algorithm.

In the following year, Choi, Wiemer-Hastings, and Moore demonstrated how to apply divisive clustering based on a Latent Semantic Analysis (LSA) from segments [CWM01]. LSA solves two main problems occurring with word frequency-based cosine similarities: short passages and synonyms.

2.2.5 Topic Modeling

Blei, Ng, and Jordan [BNJ03] introduced Latent Dirichlet Allocation (LDA) in 2003. LDA is commonly used in the literature [Eis09; Mis+09; CYY16] and can be trained on data from a single domain [Mis+11].

TopicTiling [RB12b] is an extension of Hearst 's TextTiling algorithm that uses LDA to assign topics to text blocks. Each block is represented by a T-dimensional vector, where T is the number of topics in the dataset. A coherence score is then calculated between neighboring blocks inside a "window" with cosine similarity. The depth scores of the smallest coherence scores, depending on the highest coherence scores to the left and right, indicate sub-topic boundaries.

Chen, Yao, and Yang use LDA and a K-nearest neighbor algorithm to classify short texts, which demonstrates how LDA can be applied to short text only consisting of several words [CYY16].

Tu, Xiong, Chen, and Brinton use LDA in combination with word-embeddings (cf. Section 2.3) to segment educational texts for online learning with a domain-independent algorithm [Tu+18]. They train their model on a small dataset and state that LDA can be used with a comparatively small number of topics. They also compare different similarity measures, such as cosine similarity, depth score, and spectrum. They additionally analyze the impact of different values of input parameters of LDA. A similar analysis is done by Riedl and Biemann [RB12a].

2.2.6 Keyword Extraction

Juan Ramos uses Term Frequency-Inverse Document Frequency (TF-IDF) to determine whether a word is significant to a user's query when searching documents

[Ram03]. Intuitively, a word's frequency is linked to its importance. TF-IDF proposes that not only the absolute frequency is relevant but also the number of occurrences in different documents. If a word often occurs across many documents, it is most probably not significant. The previous section describes the concept of stop words that deal with the same problem. To determine the TF-IDF, every document is run through the algorithm, and the two relevant frequencies are computed. The significance of a word is proportional to the frequency inside of the document but decreases if the word is found across different texts.

A thesaurus is another way to extract keywords [MW06] and can be especially helpful when there is only one document, and the TF-IDF approach is not suitable. A thesaurus also provides external knowledge, which allows extracting keywords without any training but requires additional maintenance and fails if no match is available.

2.3 Language Models, Word Embeddings, and Transformers

A language model estimates the probability of a piece of unseen text based on some training data [Hie09]. Modern language models allow the representation of words as continuous vectors (word embeddings). Those significantly improve and simplify many NLP applications [Mik+13]. The literature discusses different neural network architectures that estimate word vectors. The following section examines four language models: Word2Vec, ELMo, BERt, and WMT.

Word embedding is a widely used feature learning technique in NLP. The idea behind word embeddings is to represent words or phrases from a vocabulary as vectors of real numbers. Each word is represented with a point in a vector space, and the position of the point captures some of the word's semantic properties [LY17]. The feature vector captures different aspects of the word; consequently, words with the same meaning are assigned similar vector representations. These vectors can then be used as input features for various NLP tasks, such as language modeling, text classification, and machine translation. Word embeddings have proven to be an effective way to capture the meaning of words and their relationships with other words in a language.

Also, word embeddings can capture word analogies by examining various dimensions of the difference between word vectors [PSM14]. For example, the analogy "king is to queen as man is to woman" should be encoded in the vector space by the vector equation: $king - queen = man - woman$ as shown in Figure 2.1. Based on the usage of words, a distributed representation is learned that enables words used in comparable contexts to possess analogous representations. As a result, this technique naturally captures the meaning of the words.

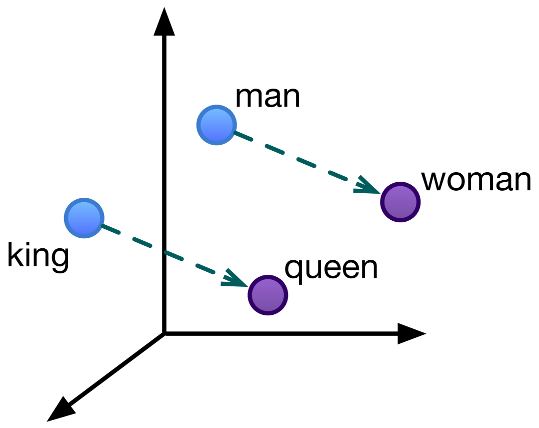


Figure 2.1: A simplified example of a three-dimensional vector space with tokens "man," "woman," "king," and "queen."

2.3.1 Word2Vec

Word2Vec is a technique to embed words into a word vector. It was proposed by Mikolov, Chen, Corrado, and Dean [Mik+13]. Word2Vec was one of the first semantically successful algorithms presenting word vectors that can catch semantic and syntactical relationships. For example, the word vector of Paris and Berlin are similar since both are capitals of a country. Word2Vec representations are learned from a simple two-layered neural network and provided state-of-the-art word embeddings [Mik+13].

2.3.2 ELMo

Embeddings from Language Models (ELMo) is a word embedding constructed as a task-specific combination of the intermediate layer representations in a bidirectional language model. ELMo maps an input sentence into a sequence of word vectors. It models complex characteristics of word use in the language. After publication, ELMo significantly improved the state-of-the-art in every considered case across a range of challenging language understanding problems [Pet+18]. As a language model, ELMo can capture both language characteristics: syntax and semantics. Most importantly, ELMo representations provide a deep contextualized word representation of word use and the variety of usages across linguistic contexts [Pet+18].

In a deep language model, the higher-level Long Short-Term Memory (LSTM) 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, as shown below, ELMo can capture both language characteristics. Figure 2.2 depicts the LSTM layers predicting the following word.

ELMo representations are characterized by three principal characteristics that allow them to achieve state-of-the-art results in most common NLP downstream tasks: Firstly, ELMo representations are contextual, wherein the representation of a given word depends upon the entire context in which it is employed. Secondly, ELMo representations are deep in nature and incorporate all layers of a deep pre-trained neural network language model. Lastly, they rely solely on character-level information, thus enabling the network to leverage morphological cues to generate robust representations for out-of-vocabulary tokens unseen in training.

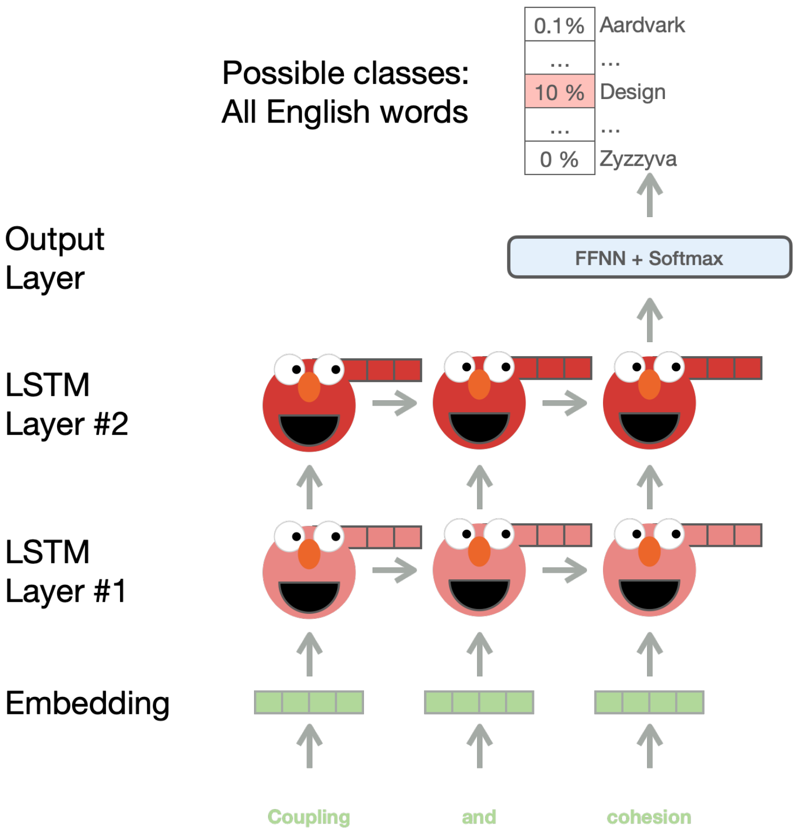


Figure 2.2: ELMo predicts the most likely following word. (Adapted from [Ala18])

2.3.3 BERT

Bidirectional Encoder Representations from Transformers (BERT) is an embedding constructed by a transformer. BERT creates the word embeddings based on the encoder of a transformer [Dev+19]. A bidirectional pre-trained model is created by masking out some percentage of the input tokens randomly and then predicting those tokens. The BERT language model enhanced the best results in several NLP tasks. [Dev+19]

2.3.4 WMT-19

The Workshop on Machine Translation (WMT)[1] is an important venue for machine translation and machine translation research, held annually since 2006. During this event, different researchers share their results and advances in machine translation during the last year. The Facebook WMT-19 model is a pre-trained model designed to translate between different languages and submitted to the WMT shared news translation task in 2019 [Ng+19].

The base system is built using the big transformer architecture [Ng+19] specified in the FAIRSEQ[2], a sequence modeling toolkit for training custom models proposed by Facebook [Ott+19].

Furthermore, WMT-19 extends the traditional transformer architecture by implementing a back translation technique, which aims to enhance the quality of machine translation systems by using data augmentation [Ng+19]. Specifically, it involves training a neural machine translation system from the target to the source language in the reverse translation direction. The system then translates target-side monolingual data back into the source language. This approach allows integrating previously unused monolingual data into the training corpus by extending the corpus with new synthetic source-to-target sentence pairs [Hoa+18]. By incorporating back translation into the training process, machine translation systems can benefit from the additional training data, improving translation quality. The WMT-19 model demonstrated the effectiveness of this approach.

[1]WMT - The Conference on Machine Translation: `https://machinetranslate.org/wmt`

[2]Facebook AI Research Sequence-to-Sequence Toolkit written in Python (FAIRSEQ): `https://github.com/facebookresearch/fairseq`

3 Clustering

Clustering, also called cluster analysis, describes the task of categorizing objects into groups (or clusters) so that objects within each group are more similar to each other than to objects outside of the group. Clustering is commonly used in various fields, including statistical data analysis, pattern recognition, machine learning, and information retrieval.

Clustering originated in anthropology research in 1932, analyzing tribal or national cultures [DK32], followed by more work in psychology in the late 1930s [Zub38; Try39; Cat43]. Since then, the field has progressed significantly. There is no ubiquitous definition of a cluster, as different measurement methods can define "similarity". Therefore, many different clustering models and algorithms exist [Est02].

In the following, we discuss clustering algorithms related to this dissertation. Section 3.1 describes partitioning-based clustering, Section 3.2 describes density-based clustering, and Section 3.3 describes hierarchical clustering. Chapter 7 [BKB21, Section "Reference Implementation"] and Chapter 8 [BKB22, Section 7] describe Athenas Clustering subsystem responsible for decomposing student answer segments into clusters.

3.1 Partitioning-Based Clustering

Partitioning-based clustering approach divides the data set into disjoint partitions and maps each data point to exactly one partition or cluster while leaving no cluster empty. K-means is a well-known and fundamental partitioning-based clustering approach. The goal of the k-means clustering is to group up n data points in k clusters [Mac67]. Its basic principle is finding k central points in the data set and assigning each point to its closest center. Finding these centers can be achieved by initializing the central points randomly and then iteratively optimizing them by computing the clusters in each step and updating the central points with the centers of the current clusters until they converge to a good enough result. An example of such an algorithm is given by Lloyd [Llo82]. K-means is a relatively fast clustering method; however, it requires the number of clusters to be specified before the start of the clustering. Dynamic forming of new clusters is not possible. Instead, Athena's clustering determines the number of clusters based on the solutions at run-time.

3.2 Density-Based Clustering

Another approach is density-based clustering [Kri+11; Cam+19], which defines clusters as regions in the data set with a higher density than the rest. Points outside dense regions can be considered outliers or noise. One of the most popular approaches is Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [Est+96]. It pre-defines a distance threshold for connecting data points and a minimum number of points to form a dense region. A dense region contains more than that minimum number of points closer to each other than the defined threshold. A cluster is defined as a dense region with all points closer than the threshold to that region. DBSCAN has two strengths: the number of clusters is determined dynamically at run time, and points in sparse regions are detected as outliers. However, a disadvantage is that the constant distance threshold makes clustering data with varying densities inaccurate. Therefore, Athena does not use density-based clustering in its pure form but in combination with hierarchical clustering.

3.3 Hierarchical Clustering

Hierarchical clustering puts the clusters in a hierarchical relationship. Depending on the concrete approach, there are two possible ways to realize this: initially specifying every data point as a cluster itself and merging them to form new clusters (agglomerative) or defining the whole data set as one cluster and dividing it to form smaller ones (divisive). The decision to form new clusters in either way depends on a metric measuring the distance between two data points. An algorithm that merges clusters in a fashion that chooses the clusters with the closest pair of points (single-linkage clustering) is SLINK [Sib73], which is optimal in both time and space. Dendrograms are helpful diagrams to analyze the results of hierarchical clustering. Athena uses the Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) [Cam+15; MH17; MHA17] algorithm, which clusters objects based on a density function. HDBSCAN is discussed in detail in Chapter 8 [BKB22].

4 Research Process & Data Collection

This dissertation is based on the design science engineering cycle, which structures the design, validation, implementation, and evaluation activities. Section 4.1 introduces design science and presents the research goals of this dissertation. The validation and evaluation activities are based on data collected in large-scale courses at the Technical University of Munich (TUM). Section 4.2 describes these courses with respect to their learning goals.

4.1 Design Science

Design science is a research approach focused on the development and validation of knowledge [Sim88] and is used in several research disciplines. Design science focuses on designing and investigating artifacts in the context of a problem domain. Design science deals with two kinds of research problems: First, designing artifacts to improve a problem context. Second, answering knowledge questions about the artifact in context. The "engineering cycle" framework is a specialization of design science for research in the domains of information systems and software engineering. The engineering cycle consists of four phases: First, the problem investigation identifies the stakeholders and their goals. Second, the treatment design specifies requirements with their contribution to the goals and available or new treatments for the problem. Third, the treatment validation investigated the effects of the treatment on the problem and whether the effects satisfy the requirements. Fourth, during the design implementation phase, the treatment is implemented. In the following iteration, an implementation evaluation replaces the problem investigation and inspects the effects of the implemented treatment and their contributions to the stakeholder goals. [Wie14]

This dissertation is based on Wieringa's design science engineering cycle [Wie14].

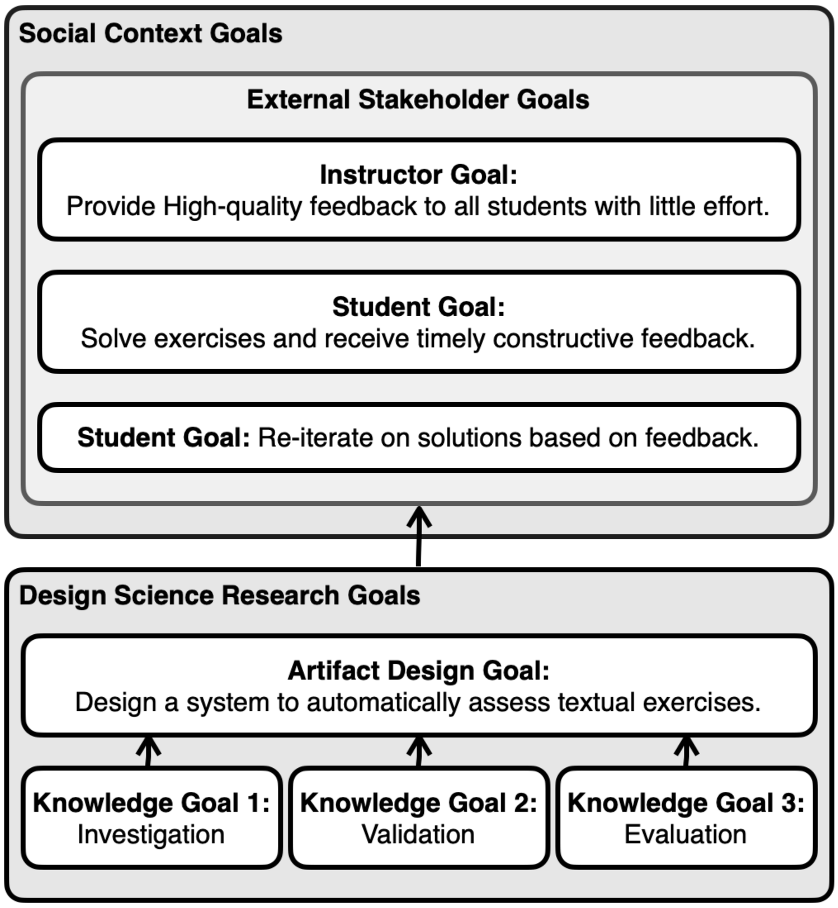

Social Context Goals
External Stakeholder Goals
Instructor Goal:
Provide High-quality feedback to all students with little effort.
Student Goal:
Solve exercises and receive timely constructive feedback.
Student Goal: Re-iterate on solutions based on feedback.
Design Science Research Goals
Artifact Design Goal:
Design a system to automatically assess textual exercises.
Knowledge Goal 1:
Investigation
Knowledge Goal 2:
Validation
Knowledge Goal 3:
Evaluation

Figure 4.1: Hierarchical goal taxonomy following the template from [Wie14]. An arrow indicates that a goal supports the other.

The two main stakeholders are instructors and students. An instructor can be either a lecturer or a teaching assistant. Lecturers are university employees such as professors, researchers, and graduate students. Teaching assistants are experienced students with knowledge in the domain and are motivated and paid to help in the teaching process. Some universities use the term "tutor" to refer to a teaching assistant. Figure 4.1 depicts the hierarchical structure of Social Context Goals and Design Science Research Goals derived from the stakeholder goals. The *design science research goals* support the *social context goals*, which in turn are defined by the *external stakeholder goals* and the *problem context*.

The instructor's goal is to deliver high-quality teaching supported by many exercises. Through individual feedback, instructors want to support students in their learning activities as much as possible. However, instructors want to minimize their workload on assessments to have time to create and improve exercises and course materials. Teaching assistants balance their limited working hours between assessments, face-to-face teaching sessions, and answering questions. The student's goal is to understand the course content and solve the exercises. Another student goal is to receive timely feedback to re-iterate their solutions and learn from their mistakes as early as possible [Pop34; Pop59].

The overall research goal of the dissertation is to reduce assessment efforts on textual exercises for instructors while providing timely feedback for students in large courses. We decompose this research goal into an artifact design goal and three knowledge goals.

To achieve the research goal, we explore ways of automating and supporting the assessment process for textual exercises with the following Artifact design goal:

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

The system consists of two artifacts: An object-oriented framework[1] "CoFee" and a reference implementation[2] called "Athena". To understand the stakeholders and the problem context, this artifact design goal is supported by the following knowledge goals 1-3:

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

To validate if Computer-aided Feedback for textual exercises (CoFee) is suited to solve the assessment problem for textual exercises, we state the second knowledge goal:

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

[2] "Object-Oriented Framework" is called "Conceptual Problem Framework" or "Treatment Design" in [Wie14].

[2] "Reference Implementation" is called "Design Implementation" in [Wie14].

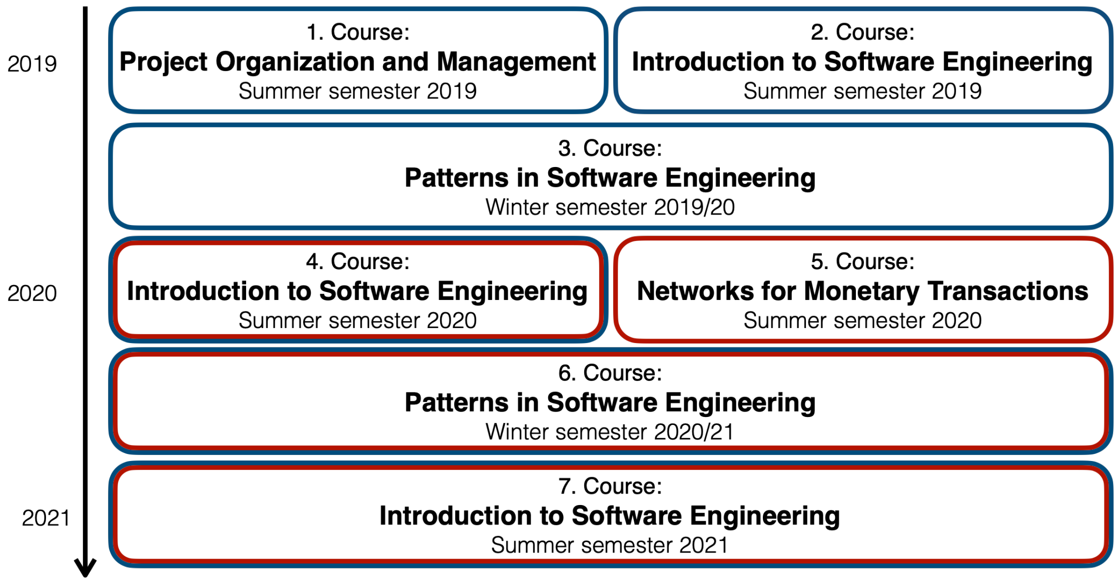

2019
2020
2021
1. Course:
Project Organization and Management
Summer semester 2019
2. Course:
Introduction to Software Engineering
Summer semester 2019
3. Course:
Patterns in Software Engineering
Winter semester 2019/20
4. Course:
Introduction to Software Engineering
Summer semester 2020
5. Course:
Networks for Monetary Transactions
Summer semester 2020
6. Course:
Patterns in Software Engineering
Winter semester 2020/21
7. Course:
Introduction to Software Engineering
Summer semester 2021

Figure 4.2: Timeline of courses used in validation and evaluation [BKB21; BKB22]. Exercises used in the courses are categorized as 🟦 Course-work and 🟥 Examination.

Athena collects assessment knowledge in the form of exercises and feedback pools. To evaluate the Athena system and its performance in large courses, we state the third knowledge goal:

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

4.2 Courses

The validation and evaluation activities are part of the design science engineering cycle and are based on exercises from seven courses at TUM. This section gives an overview of the course timeline and specifies which exercises were relevant for the evaluation. In addition, the subsections describe the courses with respect to their learning goals and the use of textual exercises in coursework and examination. Figure 4.2 visualizes the course timeline between the summer semester of 2019 and the summer semester of 2021.

Publication	Courses
[BKB21]	4. Introduction to Software Engineering (Summer term 2020)
	5. Networks for Monetary Transactions (Summer term 2020)
[BKB22]	1. Project Organization and Management (Summer term 2019)
	2. Introduction to Software Engineering (Summer term 2019)
	4. Introduction to Software Engineering (Summer term 2020)
	7. Introduction to Software Engineering (Summer term 2020)

Table 4.1: Mapping between publications and courses used as the basis for validation and evaluation.

Table 4.1 maps the courses to the publications which used data collected as part of the course. Chapter 7 [BKB21] evaluates Athena based on two courses from summer 2021. Chapter 8 [BKB22] used data from Project Organisation and Management in Software Engineering (POM) 2019/20 as part of the treatment validation. The evaluation in Publication [BKB22] is based on data from Introduction to Software Engineering (EIST) in 2020 and 2021 and makes comparisons to EIST before the introduction of Athena in 2019 as a control group. Data from Patterns in Software Engineering (PSE) 2020/21 was not used in the empirical evaluations in this dissertation.

4.2.1 Introduction to Software Engineering

The course EIST is an introductory software engineering course for computer science bachelor's students in their second semester. Students with computer science as a minor can also enroll in the course. EIST was visited by around $1,800$ students in 2020 and around $2,200$ students in 2021. Figure 1.3 depicts the development of student numbers 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 [Kru+20]. The students need fundamental programming experience as a prerequisite, typically acquired by courses such as Introduction to Informatics or Fundamentals of Programming in the first semester. The instructors use constructive alignment [Big03] to align the teaching concepts and exercises with the course objectives.

The course focuses on higher cognitive processes following Bloom's revised taxonomy [And+01]. Following an interactive learning approach, EIST teaches software engineering concepts with multiple, small iterations of theory, example, exercise, solution, and reflection [KS19].

The course involves different kinds of exercises:

1. Lecture exercises as part of the (virtual) lectures.
2. Group exercises solved in small ad hoc groups.
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 student's knowledge after the course has finished in multiple variants.

Lecture exercises motivate the students to attend the lectures [KFA17], and foster student participation [Kru+17]. The students were asked to submit their solutions to all exercises 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. To train software engineering and problem-solving skills, the instructors utilize programming, modeling, **textual**, and quiz exercises in the course.

4.2.2 Patterns in Software Engineering

The course PSE is an advanced software engineering course with around 700 computer science master's students. It teaches the principles of patterns in software development and the structure of pattern-based software systems. Students learn how to apply patterns in various problem situations and learn how to handle the patterns in concrete applications. The course covers design patterns, architectural patterns, anti-patterns, testing patterns, and pattern-based software engineering. To participate in the course, students need knowledge of object-oriented software engineering, typically obtained from the EIST course and practical software engineering courses.

Similar to EIST, PSE follows an interactive learning approach and teaches each pattern as a learning sprint consisting of theory, example, exercise, solution, and reflection [KS19]. To train understanding and application of patterns, the instructors utilize programming, modeling, **textual**, and quiz exercises in the course. These exercises are introduced as in-class exercises during the lecture where students have time to solve the exercises; however, students can finish the exercises at home. The students were asked to submit their solutions to the exercises to Artemis, where solutions are graded, and students can earn bonus points toward the final exam.

4.2.3 Project Organisation and Management in Software Engineering

The course POM teaches the most important concepts of traditional and agile software project management. Students learn to write a project plan, initiate and manage a small project, tailor a software lifecycle, and develop strategies for merge management, continuous integration, and continuous delivery.

They also get familiar with the most important risk management techniques, project scheduling, planning, testing, and software project delivery. Finally, the students apply these skills by solving simple problems in exercises and a team project.

4.2.4 Networks for Monetary Transactions

The course Networks for Monetary Transactions aims to understand and assess the fundamentals, architecture, and security of domestic and international payment networks and their legal frameworks. Around 500 students participated. The instructors used Artemis to conduct an online exam during the COVID-19 pandemic. The exam consisted of 11 quiz exercises and three text exercises. Automatic assessment suggestions based on Athena were enabled for one **textual** exam exercise: *IT-Attacks*. This exercise tests the lower-level cognitive skills and asks students to recall typical IT attack schemas.

5 Toward the Automatic Assessment of Text Exercises[1]

Publication [BB19] has been published as a **peer-reviewed workshop paper**:

Authors:	Jan Philip Bernius and Bernd Bruegge
Conference:	2nd Workshop on Innovative Software Engineering Education (ISEE '19)
Location:	Stuttgart, Germany
Pages:	19–22
Year:	2019
Review:	Peer Reviewed (4 Reviewers)
Acc. Rate:	67%

Summary The paper describes the concept for a hybrid assessment system to provide timely responses to students' homework solutions.

The system starts without any knowledge and trains its assessment model from manual assessments on the fly. It uses sentence-based grading where one piece of feedback is valid for multiple sentences from different submissions and may be shared across students.

The paper is based on the hypothesis that automated assessment can provide feedback identical to human instructors. It introduces a concept for automatically assessing text exercises using machine learning techniques. Also, it describes the plans to use this concept in a case study with 1900 students to validate the approach's applicability.

[1]This publication is embedded to provide context and to highlight the iterative research process. Publication [BB19] is not part of the evaluation of the dissertation.

Contributions **J.P. B.** developed the approach, conceptualized and formalized the scenario and workflow, outlined a potential evaluation, conducted the literature review, and wrote and visualized the paper. B. B. provided feedback and helped improving the manuscript.

Toward the Automatic Assessment of Text Exercises

Jan Philip Bernius
Department of Informatics
Technical University of Munich
Munich, Germany
janphilip.bernius@tum.de

Bernd Bruegge
Department of Informatics
Technical University of Munich
Munich, Germany
bruegge@in.tum.de

***Abstract*—Exercises are an essential part of learning. Manual assessment of exercises requires efforts from instructors and can also lead to quality problems and inconsistencies between assessments. Especially with growing student populations, this also leads to delayed grading, and it is more and more difficult to provide individual feedback.**

The goal is to provide timely responses to homework submissions in large classes. By reducing the required efforts for assessments, instructors can invest more time in supporting students and providing individual feedback.

This paper argues that automated assessment provides more individual feedback for students, combined with quicker feedback and grading cycles. We introduce a concept for automatic assessment of text exercises using machine learning techniques. Also, we describe our plans to use this concept in a case study with 1900 students.

I. Introduction and Problem

Instructors face a large population of students in their courses. Students require feedback on their exercises to reflect on their progress [1]. The concepts of interactive learning [2, 3] helps to increase the interaction between instructors and students but also increases the workload for instructors. Software engineering students need to learn constructive and creative capabilities. It is important for the instructor to facilitate the problem-solving learning process. Concrete problem-solving strategies are taught in paradigms, accepted by the profession [4]. Each paradigm provides a set of problem-solving exercises. These are usual textual exercises that involve the application of problem-solving techniques.

Exercises are a proven method to train higher cognitive skills including the acquisition of domain-specific knowledge, analysis and design methods and the evaluation of the results. Trivial exercises, such as multiple-choice quizzes, do not stimulate higher cognitive skills and do not reflect engineers daily work [1].

Exercises help students to learn, understand and apply a paradigm. A student needs feedback to reflect and improve on their solution to the exercise. Text exercise assessment causes time-intensive efforts with instructors, preventing them from spending time on improving their lectures, having discussions with their students or update exercises to incorporate technology evolution.

Increasing student populations make it harder to keep assessments fair and at equal quality. Students do not benefit from quantitative feedback alone [5]. Qualitative feedback helps students to improve.Splitting assessment efforts with multiple instructors can lead to inconsistencies. Providing timely or instant feedback in a large class is hard [6]. Waiting for feedback delays the students learning progress and hinders interactive learning. We strive toward a system to provide automated text assessments based on instructor feedback decreasing student feedback waiting times.

This paper is structured as follows: Section I introduces the domain and outlines the problems with the current correction process for text exercise. Our vision is described in Section II in the form of a visionary scenario. Section III describes the assessment workflow of a possible implementation and Virtual One-To-One, a machine learning based mechanism for providing individualized feedback for students in large classes. Section V discusses applicability and limitations of the system. We present related work in Section VI. Section IV proposes our evaluation approach, and Section VII concludes the paper.

II. Visionary Scenario

The following scenario describes how we envision to improve the assessment of text exercises:

Anna and Tom are students participating in a software engineering course. During a lecture, the instructor starts an in-class text exercise to be completed in the assessment system. Anna and Tom both submit a solution to the system. The instructor starts manually assessing a set of submissions selected by the system. The system asks the instructor to assess Annas solution. The instructor provides a score and a comment explaining his assessment. After receiving the assessment, the system decides to assess Toms solution automatically based on the assessments provided previously. Anna and Tom get individual feedback for their solution to reflect on their learning progress.

Tom is not satisfied with his submission after receiving his feedback. He decides to improve his work and resubmits a refined version of his solution. The system automatically assesses Toms resubmission and provides a new assessment. Tom is now satisfied with his assessment and fished the exercise.

III. Assessment Workflow

In a first prototypical implementation, we extend the ArTEMiS system, already capable of assessing programming and modelling exercises automatically [1, 7], by adding semi-automated text assessment. A student submits his solution for

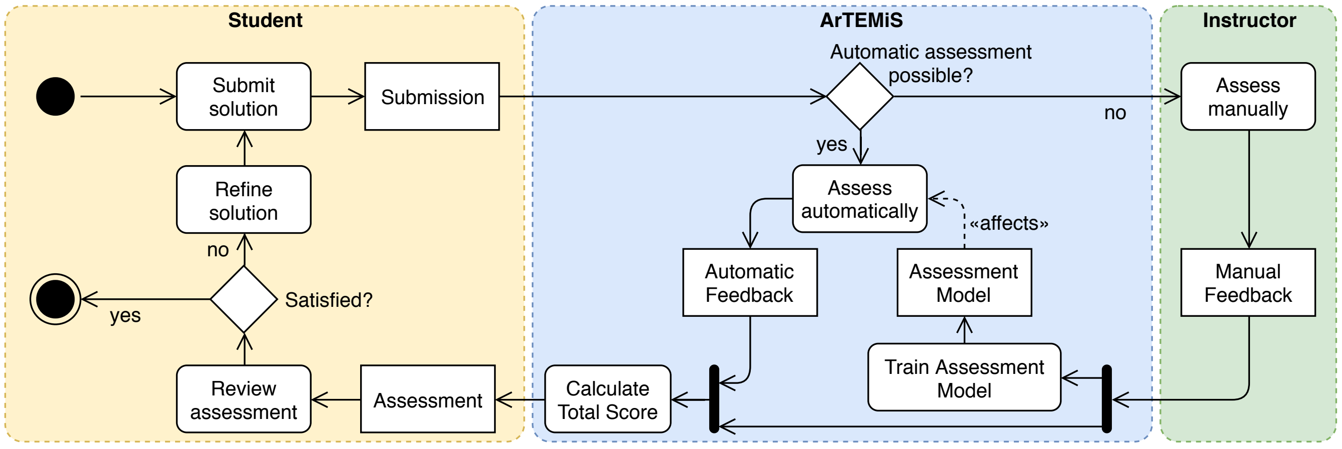


Fig. 1. Automatic assessment workflow, considering manual and automatic assessment.

a text exercise to the ArTEMiS system. The activity diagram in Fig. 1 depicts the assessment workflow. The system supports two means of assessment: Manual assessment provided by the instructor (Section III-A) and automatic assessment generated by the system based on an assessment model (Section III-B). ArTEMiS decides which assessment method is required for each submission based on the quality of the assessment model. Both means of assessment provide a set of Feedback Items.

The assessment of the submission is a composition of all feedback items. The final score is the sum of all feedback scores (see Fig. 2). Student review the assessment of their submission. If they are not satisfied, they can submit a refined solution for assessment, enabling continuous interactive learning [1] with text exercises.

A. *Manual Assessment*

ArTEMiS selects text exercise submissions for manual assessment by instructors if the assessment model does not allow for a confident assessment. Instructors are used to grading exercises using a set of rubrics. A rubric defines a set of traits of the students' submission, which are evaluated based on a scale [9]. Rubrics can exist in different levels of detail, such as only listing aspects of the assignment or defining different scoring levels. If instructors do not define a rubric beforehand explicitly, they build a rubric in their mind while assessing.

Instructors break down a submission into blocks and match each block with a rubric. As illustrated in Fig. 3, instructors define text blocks themselves as a phrase, sentence or paragraph by selecting a piece of text as they see fit. They assess each block quantitatively and qualitatively using a score and a feedback comment (see *Feedback* in Fig. 2).

B. *Automatic Assessment*

ArTEMiS assesses submissions automatically, if the quality of the assessment model allows for a confident assessment. The assessment model is trained based on the manual assessments of text blocks provided by instructors. Fig. 4 depicts the automatic assessment process. For automatic assessment, submissions need to be broken down to text blocks automatically, first. Second, a vector representation of the text blocks is calculated as an input value for further computations. Third, the assessment needs to be generated for each text block.

A first, simple approach is using sentences as text blocks. We split submissions into sentences using delimiter characters (. : ? !) or line breaks. In a later stage, we plan on applying techniques such as topic modelling for text block calculation if the simple approach does not provide sufficient results. All text blocks need feedback to complete an assessment.

ArTEMiS calculates a vector representation for each text block. Therefore, blocks are translated into a multi-dimensional vector space, following the word2vec algorithm

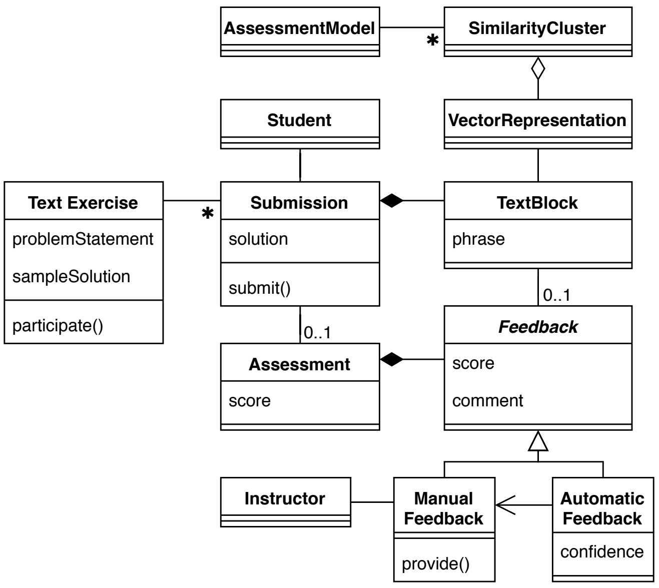
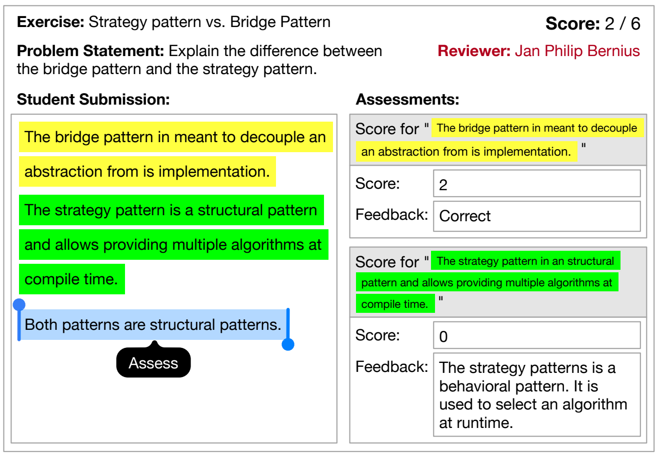


Fig. 2. The relevant entities in the system are depicted in a class diagram. A student creates a submission for a text exercise. An assessment is a composition of multiple feedback items referencing text blocks. A feedback item can be a manual or automatic feedback item. An instructor provides manual feedback. Automatic feedback items are a proxy [8] for manual feedback items. A similarity cluster aggregates the vector representations of text blocks. The assessment model consists of many similarity clusters.


Exercise: Strategy pattern vs. Bridge Pattern

Score: 2 / 6

Problem Statement: Explain the difference between the bridge pattern and the strategy pattern.

Reviewer: Jan Philip Bernius

Student Submission:

The bridge pattern in meant to decouple an abstraction from is implementation.

The strategy pattern is a structural pattern and allows providing multiple algorithms at compile time.

Both patterns are structural patterns.

Assessments:

Score for " The bridge pattern in meant to decouple an abstraction from is implementation. "

Score: 2

Feedback: Correct

Score for " The strategy pattern in an structural pattern and allows providing multiple algorithms at compile time. "

Score: 0

Feedback: The strategy patterns is a behavioral pattern. It is used to select an algorithm at runtime.

Fig. 3. Assessment of student submission for problem statement ”Explain the difference between the bridge pattern and the strategy pattern.” Example question taken from an EIST exam. Instructors define text blocks to build up their assessment. Each block is assessed with a score and a feedback text. The total score is based on all feedback items in the assessment.

[10, 11] and its doc2vec extension for sentences and paragraphs [12]. The algorithm can employ different strategies to calculate one-hot word vectors.

Using the resulting vector representation, we use cluster analysis to detect clusters of submission blocks [13] from all submissions of the same exercise. These clusters list the different statements submitted by all students as a part of their solutions.

Our primary assumption is that a single feedback item can be valid for text blocks from multiple submissions. Feedback for text blocks within the same similarity cluster can be applied to other nodes within the same cluster. This allows the system to provide VIRTUAL ONE-TO-ONE feedback: Real instructor feedback is applied to equivalent text blocks in a new submission automatically. ArTEMiS chooses a previously assessed text block located closely in the same similarity cluster, the nearest neighbour. The instructor feedback is selected for the new submission and ArTEMiS creates an automatic feedback item, a proxy for the manual feedback item (see Fig. 2).

If a cluster does not contain a manual feedback item, the system decides that an automatic assessment is not possible and requests a manual assessment from the instructor.

IV. Evaluation Approach

We plan to conduct a case study to evaluate the automated assessment quality in the *Introduction to Software Engineering* (EIST) lecture taught at the Technical University of Munich to 1900 students. Students in the course complete weekly homework exercises. We will use the system for text exercise submissions and assessments in two stages.

As the first stage, we conduct a shadow test using our prototypical implementation. The learners submit their solution to a text question using our system. Instructors establish a truth set by assessing all submissions manually. Automatic assessment is not used during this stage. The truth set will be used for quantitative evaluation of the automatic assessment accuracy by comparing automatic assessments with the corresponding manual assessment.

Hypophysis 1: Automatic assessments of text exercises following the presented concept produce results identical to manual assessments with an accuracy greater than 85%.

In a qualitative study, we will interview the instructors to analyze the block-based assessment concept (Sec. III-A), and its applicability to grading and providing feedback.

Hypophysis 2: The assessment concept allows capturing all feedback necessary for assessment of text exercises. No information is lost compared to traditional assessment.

In the second stage, we will conduct a second study in a later EIST lecture to evaluate the complete automatic assessment workflow. We will evaluate how many manual assessments are needed to generate accurate assessments and the effects on assessment time.

Hypophysis 3: Employing automatic assessment can save more than 50% in total required assessment time for all submissions. The assessment time per submission will increase compared to paper-based assessments.

A qualitative study with student interviews assesses the usefulness of automated feedback for them. Further, we want to understand students feeling toward automatic feedback.

V. Discussion

We discuss applicability, limitations and implications of automatic text assessment. Feedback generated following the concepts introduced in this paper can only be as good as the feedback provided by the instructor. The system supports the assessment process by automating the repetitive process involved in assessing text submissions.

Grading based on automatic assessment leads to ethical problems. It is unclear whether non-native language or special figures of speech could lead to decreased scores. Applications in grading should be preceded by an extensive evaluation of assessment quality. While applications in grading are out-of-scope for this paper, we propose application in a two-phase grading process only. We intend to apply the system as a learning-support system. The generated feedback should help students during their learning progress and should not be used during a grading process.

The applicability of the described systems depends on the variety of possible solutions. Exercises with a variable answer space require more knowledge for assessment, increasing the complexity. The system focuses on assessing exercises from the lower spectrum of the revised Bloom’s Taxonomy: Remember, Understand, Apply and Analyze [14]. Exercises of the given categories provide a lower variability of possible solutions and therefore limit the number of similarity clusters. Exercises from the categories Evaluate and Create are out of scope for this paper.

The design of the system allows for a hybrid assessment approach. A future system could combine manual and automatic feedback to further reduce the efforts for instructors. This could be especially useful if a certain aspect of the solution

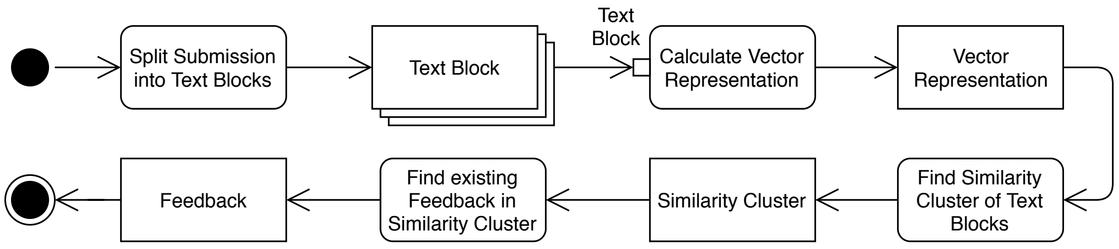

Split Submission into Text Blocks
Text Block
Text Block
Calculate Vector Representation
Vector Representation
Find Similarity Cluster of Text Blocks
Similarity Cluster
Find existing Feedback in Similarity Cluster
Feedback

Fig. 4. The automatic assessment process. Zoomed into the "Assess automatically" activity in Fig. 1.

has a larger variability. A possible example is an exercise asking for two definitions and a comparison of the terms. The variability for the definitions is small, but the variability for the comparison part is larger. A hybrid approach allows instructors to focus the manual assessment on the comparison part, as soon as the definitions can be assessed confidently.

VI. Related Work

Kiefer and Pado suggest a system to simplify the grading process presenting responses to instructors in a sorted manner [15]. Submissions are sorted by similarity with a defined sample solution. Terms used in both the sample solution and the submission are highlighted. The tool supports instructors during the grading process but does not automatically assess submissions. The only criterion is the sample solution. Instructor assessments are not considered for the following submissions.

Wolska et al. and Basu et al. suggest a grading process where instructors grade submissions sorted by clusters of similar submissions for exercises in the domains of German as a foreign language [16] and the United States Citizenship Exam [17]. They propose clusters of entire submissions, compared to the text block based clustering approach presented in this paper. Basu et al. introduce grading of an entire cluster of submissions as a single action [17].

Gradescope Inc. offers its tool Gradescope, a commercial solution for grading assistance and "AI-assisted Grading". Their core product offers a rubric based grading system, allowing instructors to define a set of scores with feedback comments per exercise. Instructors manually select rubrics for each submission. Changes to the scores and comments in a rubric are applied to previously assessed submissions. The "AI-assisted Grading" feature creates groups of submissions (compare with similarity clusters), allowing the instructor to select rubrics for the entire group of submissions, similar to the approach of Basu et al. [17]. The automatic creation of groups is limited to multiple-choice and fill-in-the-blank exercises. It does not offer an automatic grouping of text questions.

These works focus on traditional exam assessment. The primary objective is an accelerated grading process, rather than providing feedback through comments. The focus of our approach is primarily providing more qualitative feedback to students on homework and in-class assignments.

VII. Conclusion

Assessments of text exercises require time-intensive efforts from instructors today. We argue that an automated process to generate Virtual One-To-One feedback can reduce assessment efforts for instructors and increase the amount of feedback for students. The system should use machine learning techniques to detect text blocks of the same meaning in submissions and automatically link real instructor feedback to equivalent blocks.

References

[1] S. Krusche and A. Seitz, "Increasing the Interactivity in Software Engineering MOOCs - A Case Study," in *31th Conference on Software Engineering Education and Training*, 2019.
[2] D. Kolb, *Experiential Learning: Experience As The Source Of Learning And Development*. Prentice Hall, 1984, vol. 1.
[3] S. Krusche, A. Seitz, J. Börstler, and B. Bruegge, "Interactive Learning: Increasing Student Participation through Shorter Exercise Cycles," in *19th Australasian Computing Education Conf.* ACM, 2017, pp. 17–26.
[4] T. S. Kuhn, *The Structure of Scientific Revolutions*. University of Chicago Press, 1996.
[5] P. Sadler and E. Good, "The Impact of Self- and Peer-Grading on Student Learning," *Educational Assessment*, vol. 11, no. 1, pp. 1–31, Feb. 2006.
[6] G. Jerse and M. Lokar, "Providing Better Feedback for Students Using Projekt Tomo," in *1st ISEE Workshop*, 2018, pp. 28–31.
[7] S. Krusche and A. Seitz, "ArTEMiS - An Automatic Assessment Management System for Interactive Learning," in *49th Technical Symposium on Computer Science Education*. ACM, 2018.
[8] B. Bruegge and A. Dutoit, *Object-Oriented Software Engineering Using UML, Patterns, and Java*, 3rd ed. Prentice Hall, 2009.
[9] V. J. A. Barbara E. Walvoord, *Effective Grading: A Tool for Learning and Assessment in College*, 2nd ed. Jossey-Bass, 2009.
[10] J. Mitchell and M. Lapata, "Vector-based Models of Semantic Composition," in *46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 2008, pp. 236–244.
[11] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," *CoRR*, vol. 1301.3781, 2013.
[12] Q. Le and T. Mikolov, "Distributed Representations of Sentences and Documents," in *31st International Conference on Machine Learning*, vol. 32, 2014, pp. II–1188–II–1196.
[13] N. Bansal, A. Blum, and S. Chawla, "Correlation Clustering," *Machine Learning*, vol. 56, no. 1-3, pp. 89–113, Jul. 2004.
[14] D. Krathwohl, "A revision of bloom's taxonomy: An overview," *Theory into Practice*, vol. 41, no. 4, pp. 212–218, 2002.
[15] C. Kiefer and U. Pado, "Freitextaufgaben in Online-Tests – Bewertung und Bewertungsunterstützung," *HMD Praxis der Wirtschaftsinformatik*, vol. 52, no. 1, pp. 96–107, Feb. 2015.
[16] M. Wolska, A. Horbach, and A. Palmer, "Computer-Assisted Scoring of Short Responses: The Efficiency of a Clustering-Based Approach in a Real-Life Task," in *Advances in Natural Language Processing*. Springer, 2014, pp. 298–310.
[17] S. Basu, C. Jacobs, and L. Vanderwende, "Powergrading: a Clustering Approach to Amplify Human Effort for Short Answer Grading," *Transactions of the Association for Computational Linguistics*, vol. 1, pp. 391–402, 2013.

6 Towards the Automation of Grading Textual Student Submissions to Open-ended Questions

Publication [Ber+20] has been published as a **peer-reviewed conference paper**:

Authors:	Jan Philip Bernius, Anna Kovaleva, Stephan Krusche, and Bernd Bruegge
Conference:	4th European Conference of Software Engineering Education (ECSEE '20)
Location:	Seeon, Germany
Pages:	61–70
Year:	2020
DOI:	10.1145/3396802.3396805
Review:	Peer Reviewed (4 Reviewers)
Acc. Rate:	69%

Summary It proposes a support system for grading textual exercises using an automatic segment-based assessment concept. The system provides suggestions for instructors by reusing feedback from previous submissions.

This paper presents the design and a prototypical implementation of an algorithm for segment-based grading. Delimiting by punctuation marks is not a viable solution to derive segments due to students' inconsistent use of punctuation. Instead, it uses topic modeling to decompose the student solutions into smaller units. The algorithm captures the meaning of segments in keywords and applies a divide-and-conquer strategy to combine atomic text segments between topic shifts. The system produces small units for assessment, generating reusable and structured feedback templates for instructors.

The algorithm was evaluated qualitatively by comparing automatically produced segments with manually produced segments created by humans. The results demonstrate that the system can produce topically coherent segments that can be used in the assessment process.

Contributions **J.P. B.** initiated the idea of segmenting student answers using topic modeling techniques. A. K. conducted the literature review. A. K. and **J.P. B.** conceptualized and modeled the algorithm. A. K. implemented the system. **J.P. B.** predominantly wrote the article with the support of A. K. **J.P. B.** visualized the paper. B. B. provided feedback and helped improving the manuscript. S. K. and B. B. reviewed the paper.

Conference paper The author's version of the conference paper is reprinted in this dissertation. The final version of record is available at:
`https://dl.acm.org/doi/10.1145/3396802.3396805`

Towards the Automation of Grading Textual Student Submissions to Open-ended Questions

Jan Philip Bernius
Department of Informatics
Technical University of Munich
Munich, Germany
janphilip.bernius@tum.de

Anna Kovaleva
Department of Informatics
Technical University of Munich
Munich, Germany
anna.kovaleva@tum.de

Stephan Krusche
Department of Informatics
Technical University of Munich
Munich, Germany
krusche@in.tum.de

Bernd Bruegge
Department of Informatics
Technical University of Munich
Munich, Germany
bruegge@in.tum.de

ABSTRACT

Growing student numbers at universities worldwide pose new challenges for instructors. Providing feedback to textual exercises is a challenge in large courses while being important for student's learning success. Exercise submissions and their grading are a primary and individual communication channel between instructors and students. The pure amount of submissions makes it impossible for a single instructor to provide regular feedback to large student bodies. Employing tutors in the process introduces new challenges. Feedback should be consistent and fair for all students. Additionally, interactive teaching models strive for real-time feedback and multiple submissions.

We propose a support system for grading textual exercises using an automatic segment-based assessment concept. The system aims at providing suggestions to instructors by reusing previous comments as well as scores. The goal is to reduce the workload for instructors, while at the same time creating timely and consistent feedback to the students. We present the design and a prototypical implementation of an algorithm using topic modeling for segmenting the submissions into smaller blocks. Thereby, the system derives smaller units for assessment and allowing the creation of reusable and structured feedback.

We have evaluated the algorithm qualitatively by comparing automatically produced segments with manually produced segments created by humans. The results show that the system can produce topically coherent segments. The segmentation algorithm based on topic modeling is superior to approaches purely based on syntax and punctuation.

CCS CONCEPTS

• **Social and professional topics** → **Software engineering education**; • **Computing methodologies** → *Natural language processing*.

KEYWORDS

Software Engineering Education, Automatic Assessment, Textual Exercise, Assessment Support Systems

ACM Reference Format:
Jan Philip Bernius, Anna Kovaleva, Stephan Krusche, and Bernd Bruegge. 2020. Towards the Automation of Grading Textual Student Submissions to Open-ended Questions. In *European Conference on Software Engineering Education (ECSEE '20), June 18–19, 2020, Seeon/Bavaria, Germany.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3396802.3396805

1 INTRODUCTION

In the past, there has been a growing number of students enrolled at universities worldwide[1]. Large courses have thousands of students participating, especially when using virtual classrooms. Figure 1 shows a typical mixed classroom setup for 1.700 software engineering students used in the summer semester 2019 at Technical University of Munich (TUM).

In introductory computer science and software engineering courses, classroom sizes with up to 1.700 students are no longer an exception, with growth by factor five in the last ten years. The free Stanford Massive Open Online Course (MOOC) "Intro to Artificial Intelligence,"[2] started in 2011, quickly reaching 160,000 students [42]. Large lectures pose a problem for instructors when grading textual exercises. This is partially solved in MOOCs by peer reviews [19]. The main problem is the asynchronous assessment, which usually requires a week, or even longer. A major disadvantage of MOOCs is the delay between giving the exercise and grading. To reduce this delay, we teach interactive lectures where we include exercises live during the lectures, grade them immediately, and provide quick feedback to students [24]. This increases student comprehension and deepens understanding [19, 24], "significantly by up to 87 %" in the domain of modeling [25].

Technology to foster interaction and discussion within large lectures does exist [19, 29], as well as a scalable exercise system for programming and modeling exercises with automatic assessments

[1]United Nations, "UN Global Assessment on Higher Education Reveals Broad Socio-Economic, Gender Disparities," https://news.un.org/en/story/2017/04/555642-un-global-assessment-higher-education-reveals-broad-socio-economic-gender, 2017.
[2]Peter Norvig and Sebastian Thrun, "Intro to Artificial Intelligence," https://www.udacity.com/course/intro-to-artificial-intelligence--cs271, 2011.

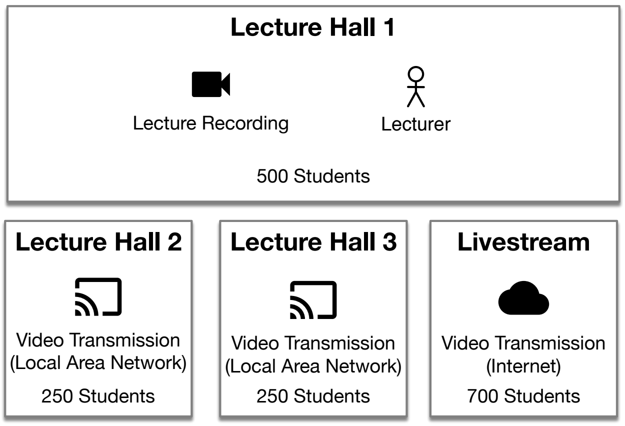


Figure 1: Mixed on-campus and virtual classroom setup employed in the summer semester 2019 at TUM for the "Introduction to Software Engineering" course.

[22, 23]. Textual exercises are commonly used in the examination, but no automatic assessment solution is available to instructors.

Conducting open answer questions requires time-consuming activities from instructors, including designing exercises and manual assessment, due to the high variability in student answers. To reduce efforts, instructors tend to reuse exercises from previous years. Grading is a repeatable process, instructors look for common mistakes or predefined solution patterns. The students learning success benefits from detailed and personalized feedback [37]. To enable large scale courses, the need to reuse feedback comments arises. Individual feedback can still rely on the domain expertise of the teacher. A single instructor cannot provide regular individual feedback due to large student bodies with more than 1.000 students. Ofter, tutors are employed to distribute the workload. Multiple graders require means to create consistent feedback for learners. This holds especially if the assessments are relevant for the final grade, e.g. as part of a grade bonus system.

This paper focuses on the segmentation of submissions into topically coherent parts, to enable reuse of feedback. Section 2 describes foundations on assessment systems and Section 3 summarizes related work on text segmentation. We present an algorithm in Section 4 that learns the topics of the submissions and then splits up the answers accordingly. The Evaluation in Section 5 analyzes the quality of the algorithm's performance, in a study with 10 participants. Section 6 summarizes the paper and outlines future work.

2 ASSESSMENT SYSTEMS

Assessment systems are a common tool used in universities. Software systems available to instructors vary from simple submission of work, over grade review, towards automated systems. We first explore interactive learning, a teaching methodology that can be supported by assessment systems. Second, we inspect Artemis as an example of an assessment system geared towards automatic assessment. Last, we look at an approach to apply automatic assessments on textual exercises.

2.1 Interactive Learning

A traditional university approach based on real-time communication demands students to be present in the lecture hall to participate. With growing numbers of enrollments in universities, the interaction in classes is getting more difficult as more staff is needed, and new ways for communication in large audiences are required [19]. One of the first approaches to incorporate technology into the classroom was the introduction of clickers for answering questions [29]. Mayer et al. describe a method for forcing interaction with the help of "response systems". The proposed system allows students to "click" an answer to a multiple-choice question. The instructor can evaluate the answers and a discussion on the topic can follow. Bonwell and Eison analyze the impact of in-class discussions and questions during lectures and exercises [6]. They found out that through constantly applying knowledge, students gain a deeper understanding of the content. The interactive learning approach combines theory, typically presented in lectures, with practical exercises [23]. Reflections based on feedback help to comprehend knowledge. In an iterative process, frequent feedback enables students to resubmit and learn from their mistakes [24].

2.2 Artemis

Artemis[3] is an automatic assessment management system developed at TUM [22]. It was built specifically to enable interactive lectures, following the idea of interactive learning. The aim of this system was primarily to allow students that are enrolled in software engineering classes to participate in interactive programming exercises. The system provides quick automatic feedback, thereby helping the students to acquire knowledge better and, as a result, achieve better grades in the final exam [24, 25]. During the past years, the system constantly evolved and is now also used at other educational institutions and in MOOCs. Programming exercises can be submitted and assessed with the help of unit tests. Additionally, modeling exercises are supported by a UML editor and a semi-automatized assessment component. The system provides full support of multiple-choice quizzes, including creating, conducting, and correcting them. For a deeper understanding of a lecture's theoretical basis, open-ended questions are more suitable than multiple-choice questions [13]. Artemis allows us to conduct textual exercises and submit answers, but instructors need to grade student answers manually. This is a time-consuming process that can lead to longer feedback loops, which decrease the students' motivation. With a growing number of students, the number of assessments increases, too. This results in bigger workloads for instructors and usually requires hiring more people. In this case, the consistency of the assessment may decrease. While there is usually only one sample solution, an unbound variety of students' answers exists. In mathematical problems or multiple-choice questions, the correct solution is mostly unique, whereas, for open questions, multiple interpretations are possible.

[3]"Artemis: Interactive Learning with Individual Feedback," https://github.com/ls1intum/Artemis, 2020.

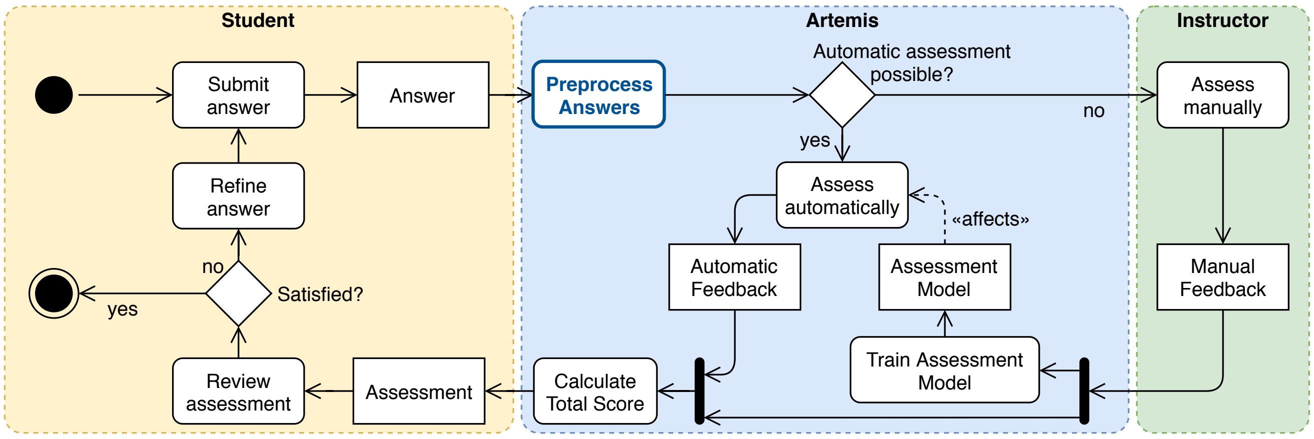


Figure 2: Workflow of the automatic assessment system for textual exercises. The "Preprocessing Answers" activity (Figure 4) includes the algorithm presented in this paper. UML activity diagram based on Bernius and Bruegge [2].

2.3 Automatic Assessment of Textual Exercises

Bernius and Bruegge describe a feedback concept built to produce reusable and consistent feedback targeted for automatic assessments of textual exercises [2]. Feedback is provided to topically coherent text blocks, resulting in uniform and consistent feedback across all assessments from multiple instructors. The concept aims at reducing work for instructors and increasing consistency, reducing complaints from a peer-to-peer comparison between students. In this approach, text blocks are manually highlightable by the instructor, but this is not applicable to automated computations. Splitting student answers based on delimiter characters[4] is not a reliable solution, because of missing punctuation, abbreviations, the use of bullet point answers, or long sentences. Also, a single feedback item is sometimes more suitable for a whole paragraph or a single clause or bullet point, which is not covered by the syntactical separator approach and requires manual adjustments.

Based on this concept, we developed a system to reuse instructor feedback across students by analyzing the similarity of text blocks [2]. The system simplifies the grading process by providing grading suggestions to instructors. Feedback suggestions are based on similarity between answers, allowing the training of an assessment model used to automatically assess answers as depicted in Figure 2. Training and using this system relies on topically coherent text blocks so that feedback is well scoped and can be shared between many submissions.

3 TEXT SEGMENTATION

Text segmentation is considered to be one of the tasks of Natural Language Processing (NLP). The term is used differently in literature and is not clearly defined. For example, document processing to extract typed or handwritten text by distinguishing it from graphics and blank spaces is referred to as text segmentation [17]. In other cases, text segmentation is the process of extracting text from video in order to index the recordings in a database [26]. Pak and Teh conducted an analysis of literature on text segmentation published between 2007 and 2017 [34] and categorize different approaches found in literature as depicted in Figure 3. The authors additionally categorize the papers according to used documents, language, and the goal of applying text segmentation. They identify the following application domains for text segmentation: "emotion extraction, sentiment mining, opinion mining, topic identification, language detection and information retrieval" [34].

[4]Delimiter characters such as . : ; ? !

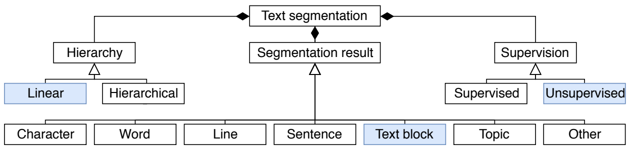


Figure 3: Taxonomy for text segmentation adapted from Pak and Teh [34]. Text segmentation types relevant for this paper are highlighted in blue color.

Information retrieval has many different applications, for example, reducing large documents to relevant fragments based on desired subtopics. The different desired results of text segmentation, the segment, is another interesting aspect Pak and Teh point out. According to their analysis, a word is considered a segment most often in literature, slightly less frequent are characters, topics, sentences, lines. In other cases, phrases, paragraphs, or tags can be used. We define the term "text block" in this paper as either clause, bullet point, sentence, or paragraph.

Text segmentation can be additionally divided into linear, text split into non-overlapping linear segments, and hierarchical, where segments also have hierarchical relationships [9, 44]. The latter is sometimes used for discourse retrieval. Along with most literature on text segmentation, we only focus on linear text segmentation.

There also exists a differentiation based on the supervision of the algorithm. Unsupervised approaches do not require any external information to be trained, whereas supervised algorithms learn from big datasets, such as Wikipedia, for example [21].

3.1 Topic Modeling

Latent Dirichlet Allocation (LDA) was introduced by Blei et al. in 2003 [5]. LDA is used by many authors [7, 9, 32, 33] and proven to be suitable when training data is from the same domain as test data [32].

TopicTiling is an extension of Hearst's TextTiling algorithm that uses LDA to assign topic IDs to text blocks [15, 41]. Each block is represented by a T-dimensional vector, where T is the number of topics in the dataset. A coherence score is then calculated between neighboring blocks inside of a "window" with cosine similarity. Depth scores of the smallest coherence scores are then calculated depending on the highest coherence score to the left and the right. The highest depth scores indicate sub-topic boundaries.

Chen et al. use LDA and a K-nearest neighbor algorithm to classify short texts which gives evidence that LDA can also be applied to text consisting of only several words [7].

Tu et al. use LDA and word-embeddings to segment educational texts for online learning with a domain-independent algorithm [43]. They train their model on a small dataset and state that LDA can be used with a comparatively small number of topics. They also compare different similarity measures, such as cosine similarity, depth score, spectrum. They additionally analyze the impact of different values of input parameters of LDA. A similar analysis is done by Riedl and Biemann [40].

3.2 Keyword Extraction

Ramos uses Term Frequency Inverse Document Frequency (TF-IDF) to determine whether of a word is significant to a user's query when searching documents [38]. Intuitively, a word's frequency is linked to its importance. TF-IDF proposes that not only the absolute frequency is relevant, but also the number of occurrences in different documents. If a word occurs often across many documents, it is most probably not significant. In the previous section, the concept of stop words, which deals with the same problem, is described. The application of TF-IDF is rather straightforward: every document is run through and the two relevant frequencies are computed. The significance of a word is proportional to the frequency inside of the document but decreases if the word is found across different texts.

Another way to extract keywords is by using a thesaurus [30]. This can be especially helpful when there is only one document, thus, the TF-IDF approach is not suitable. A thesaurus also provides external knowledge which, on one hand, allows extracting keywords without any training but, on the other hand, requires additional maintenance and fails if there is no match available.

An ontology, a relational representation between concepts, can also be used to extract topics from text [11]. Embley et al. take unstructured documents and application ontology as input. Then they use a "keyword recognizer" to spot keywords with the help of regular expressions, afterward, restructuring the extracted information with the help of the ontology. They use this approach, for example, to extract information from car advertisements. This can be a suitable solution if the domain is known, keeping in mind that creating an ontology requires time. However, it is not applicable if the algorithm is to be applied to many different domains, and the main concepts are not known in advance.

Matsuo and Ishizuka propose another method for keyword extraction that bases on the χ^2-measure [28]. They first count co-occurrences of words and word sequences. "If a term appears frequently with a particular subset of terms, the term is likely to have an important meaning" [28]. Then a co-occurrence matrix is calculated. To improve the χ^2-computation "variety of sentence length and robustness of the χ^2-value" are considered. To improve the quality of the χ^2-measure two types of clustering are applied. Similarity-based clustering gathers words with similar roles in a sentence, pairwise clustering picks words from the same domain. The words with the largest χ^2-value are given as the result.

Most of the previous approaches only focus on the frequencies but cannot detect synonyms, even different forms of a verb can decrease the quality of the algorithms. Hulth adjusts the previous approaches by introducing syntactical information, such as part-of-speech (PoS) tagging, and data preprocessing, for example, stemming, stop words removal [16]. They introduce a pattern approach: based on the training set, there is evidence that most keywords have nouns and follow a particular pattern, for example, "adjective noun" uncountable or in the singular [16]. To calculate the relevance of a phrase four features are used: frequency within a single document, frequency in the whole set of documents, the position where the term appears first in a document, and the PoS-tag. The machine learning model is then based on a set of inductive rules that are derived with the help of "recursive partitioning (or divide-and-conquer), which has as the goal to maximize the separation between the classes for each rule" [16].

3.3 Dataset

Most authors use labeled and segmented, often artificially generated datasets, such as Choi's labeled dataset for evaluating their algorithms [8]. Often news articles or news broadcast transcripts are used as there are clear topic boundaries that can be then compared [1, 7, 20, 35, 45]. The evaluation algorithm is often based on the approach by Beeferman et al. [1, 12, 39].

In this paper, as part of a text grading application for a university environment, we focus on data collected from the lecture "Patterns in Software Engineering" (PSE) at TUM in 2018/19. The dataset consists of two exercises with 121 and 124 student submissions. The exercises were conducted in-class and were announced as a mock exam.

4 SEGMENTING STUDENT ANSWERS

Based on the literature, several existing approaches were applied to the proposed problem. For testing the approaches we used a set of students' answers from our dataset. The exercise on the difference between patterns and anti-patterns received answers with an average length of 3.6 sentences. Tested approaches were, TopicTiling [41] and Bayes-seg developed by Eisenstein and Barzilay [10]. The first is based on topic modeling with LDA. The latter uses Bayesian probabilities and entropy to segment the texts. However, these algorithms could deliver no or only poor results on our dataset, most probably because of the short length and specific vocabulary distribution, which they are not fitted for.

We abstracted the topic modeling approach and preserve the idea that every answer is a collection of topics, and many topics

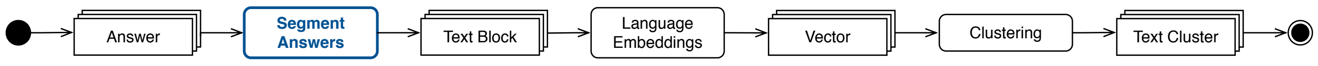


Figure 4: A detailed view into the "Preprocess Answers" activity (Figure 2) performed by the assessment system before the grading, depicted using a UML activity diagram.

are distributed among different answers [5]. However, instead of calculating a topic model, we claim that a topic can be reduced to a keyword. This way, the scarcity of the words in the answers can be compensated for. Another strategy adapted from other works is the "vocabulary introduction" [15]. As soon as new keywords are introduced, a new segment begins. The presented approach differs from thesaurus or ontology in a way that we do not know what the keywords are going to be, and they are calculated for every problem separately.

In the assessment system, the algorithm is one step in a preprocessing phase, depicted in Figure 4. Answers are segmented into text blocks before language embeddings and clusters are computed.

The algorithm can be separated into three phases: Text Preprocessing, Keyword Extraction, and Segmentation. Figure 5 depicts the algorithm's flow of events, which is described in detail in the following sections.

4.1 Text Preprocessing

Most algorithms for NLP are applied to preprocessed text-data. In the assessment context, data is of rather low quality and cannot be preprocessed manually. The available data contains lots of typing mistakes, poor formatting, missing punctuation, and misspelled words. Student submissions must not be modified, formatting being the only exception. Applying existing algorithms to our data showed that bullet points, wrong punctuation, such as using newlines instead of points, can quickly reduce the quality of the outcome. Hence, we try to cover the most common irregularities and transform them into a format suitable for further calculations.

4.1.1 Stop Words. Removing stop words from text is a very common way to clean textual data for NLP [15, 16, 41]. We use the set of stop words provided as part of the Natural Language Toolkit for Python (NLTK) [4]. The English collection consists of 179 words, like "I", "the", "what", "did", that do not contain much lexical content and can, therefore, be removed from the corpus. Although this implementation only supports students' submissions written in English, the German set of 232 words is also included because occasionally students hand in answers in the German language. This cannot provide full support of submissions in German but can reduce their negative effect on further processing.

4.1.2 Lemmatization. Lemmatization is the process of reducing a word to its meaningful root. Keeping in mind, that we want to extract keywords from a text and that the stop words are already removed, we now have a set of words where the most significant terms need to be found. Naturally, we use different forms of a word: either the plural or the singular, different tenses for verbs, degrees of comparison for adjectives, etc. Without preprocessing, the system would consider the words "view" and "views" as two different ones. With the help of WordNet, which is provided as part of the NLTK, the algorithm reduces the second word to "view" [4, 31].

The result of the text preprocessing is thereby a set of lemmatized lower-case words without any punctuation or stop words.

4.2 Keyword Extraction

The chosen approach for segmenting the students' answers into text blocks is partially based on keyword extraction. We generalize the idea of topic modeling that claims that every document is a distribution over topics, and every topic is distributed over words. We claim that every student's submission is a collection of topics, and statements, that are common among different answers. However, we do not calculate a topic model. As already described, existing approaches based on topic modeling are not suitable for our kind of data because of rather short answers (3.6 sentences long on average) and very different vocabulary used among different submissions. That is why we reduce a topic to a keyword, thereby, compensating for the data scarcity.

For keyword calculation, we adopt an approach based on word frequency[5]. We tested the frequently used TF-IDF approach [38], which proved to be inefficient in our case. The reason for it is the specific character of the data. The TF-IDF method assumes that words, frequent among different documents, are not significant for keyword extraction, as they are too common. In the considered context, the important words, definitions, for example, are present in most of the answers. Another examined approach was an extension of the word frequency measure [16, 39]. Instead of searching for significant words, they consider n-grams. This method did not suit the data either. We tested the algorithm with bi- and tri-grams, the resulting segmentation was worse than with single words. The resulting keywords are the 10 most frequently used words in the texts. The number was chosen empirically based on our data. Dynamically determining the optimal number of keywords could be researched in the future to improve the algorithm.

4.3 Segmentation

The segmentation of the texts is split up into two steps. First, the answers are split up into initial text blocks. Then, adjacent text blocks are considered and merged if there are no new keywords introduced. The result of this is a set of segments for each answer that can be used by the rest of the system.

4.3.1 Sentence Tokenization. For identifying sentences we use a pre-trained model of "punkt tokenizer" from the NLTK [4, 18]. However, it cannot handle bulleted lists, that is why we need to additionally split the text on new lines. We also want to work with clauses if a sentence is long. We decided not to use any algorithm for that but search for conjunctions. We use subordinating conjunctions and assume that they indicate a new clause. This approach is not

[5]Sowmya Vivek, "Automated Keyword Extraction from Articles using NLP", https://medium.com/analytics-vidhya/automated-keyword-extraction-from-articles-using-nlp-bfd864f41b34, 2018.

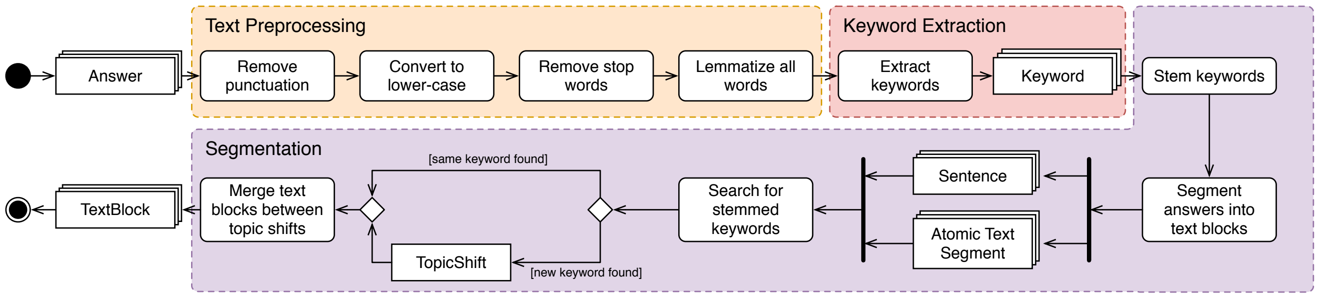


Figure 5: The segmentation algorithms flow of events depicted using a UML activity diagram based on Bernius et al. [3].

complete and cannot be considered proper clause identification, however, for this use case, we assume, it is enough. To minimize false positives when identifying clauses, we only consider sentences that are longer than 20 words.

4.3.2 Finding Segments. Before searching for keywords in the text blocks, we use a stemmer from the NLTK [4], called PorterStemmer [36]. Similarly to lemmatization, stemming is applied to avoid different forms of a word in a text. The latter, however, reduces a term to a part, that in some cases may not be a correct word. An example of this is "similarit", as the result of stemming the word "similarity". Hence, it can be very helpful when searching for words, as you can then find both "similarities" and "similarity".

For the definition of segments, we use the lexical cohesion approach and the vocabulary introduction method [14, 15]. The algorithm iterates over all text blocks defined in a submission. We use the original texts at this stage, not the preprocessed versions. In every segment, stemmed keywords are sought. If two adjacent segments have the same keywords or the second text block has none, they are merged into one block and the algorithm proceeds. As soon as new keywords are introduced, the algorithm puts a segment boundary before the current text block. This way the whole process can be defined as a "divide & conquer" approach, because we first divide the answer into initial text blocks, as small as possible, and then merge them according to the defined boundaries.

5 EVALUATION

In order to evaluate the segmentation quality of the algorithm, we conducted a qualitative study with 10 participants. We compare the segmentation of the new algorithm with the existing approach and the segmentation generated by the participants. We present anecdotal evidence on the performance of the new algorithm.

5.1 Design

The evaluation is designed as a 15-minute interview. Participants first get an introduction to semi-automatic text assessment, the assessment concept [2] and segmentation. However, for reasons of internal validity, no details of the segmentation algorithm or further processing are given. The questionnaire consists of two parts: segmentation tasks and questions about the subjective impressions of the approach.

The first part requires five segmentation tasks. Participants are given five student submissions from our dataset and asked to find and mark topic shifts. The same task is performed by two systems, one based on the syntactical separator approach and a second one based on our topic modeling algorithm.

Each participant performs the task of finding and marking topic shifts, as the system would do. These results are then quantitatively analyzed and compared to the segmentation results of the existing solution and the proposed algorithm. The performance measure consists of the two criteria recall and precision [1]:

$$recall = \frac{\text{number of estimated topic shifts that are actual topic shifts}}{\text{number of true topic shifts}}$$

$$precision = \frac{\text{number of estimated topic shifts that are actual topic shifts}}{\text{number of estimated topic shifts}}$$

The submissions are taken from the PSE dataset and are of various format that is common among students' answers. There are bulleted and numbered lists, as well as text mixed with bulleted lists, also two submissions that consist of multiple sentences and paragraphs are included. The submissions are taken with original grammar and punctuation.

The third part addresses the impressions of the surveyed. They are asked to state their personal opinion on the approach and give their judgment whether this solution can improve the instructors' and students' experience with textual exercises. The possible answers are on a five-point scale based on Likert [27].

The study was conducted with ten students from the Department of Informatics at TUM, who previously passed software engineering courses from our chair four of which have previous experience working as a tutor. These students have reasonable domain knowledge to determine segments. Also, they are potential tutors for future editions of the courses.

5.2 Objectives

We define the following hypotheses for the evaluation:

H1 The designed segmentation algorithm performs better than the syntactical separator approach measured using the performance criterion recall and precision.
H2 Students understand the approach and find it intuitive.
H3 Students consider the approach an improvement of their understanding of feedback and the comprehension of a lecture's content.

H4 The segmentation algorithm produces the same segmentation as humans.

5.3 Results

Based on the computed segmentations depicted in Figure 6, we conducted a performance evaluation using recall and precision. A topic shift position was considered if more than 50% of the students marked the position. Results in Table 1 show an increased recall and precision values for the topic modeling based algorithm.

Table 1: Performance analysis of the new topic modeling based algorithm and the previous approach based on syntactical separators measured according to precision and recall [1].

Submission	Topic Modeling		Syntactical Separators	
	Recall, %	Precision, %	Recall, %	Precision, %
S1	100	100	100	50
S2	75	60	100	67
S3	75	100	50	50
S4	100	100	100	100
S5	67	100	30	100
Average	83.4	92	76	73.4

We analyze the number of detected topic shifts in Figure 7. We compare the number of topic shifts found by the proposed algorithm and the current solution to the number of topic shifts marked by the participants. We also depict statistics for the most frequent topic shifts, meaning positions that were present in six or more answer sheets.

In the questionnaire, nine out of ten students agreed that the presented approach of segmenting answers is intuitive (see Figure 8), supporting our hypothesis H2. Students also claimed that finding topic shifts' positions was not very easy which can probably be linked to the unambiguity of the task. The results also depend on the style of the assessment of a participant. Therefore, we compared the average number and the number of the most frequent segments, where one can see that these two numbers sometimes vary. Especially, for submission S2, where the proposed system failed to improve the result of the current system, the difference between the two numbers is big. This can also be justified with the fact, that some participants tended to mark more positions than other students for most of the submissions. The data shows that the topic modeling-based algorithm resembles human perception better than the syntactical separation approach.

Since most of the students stated to value the assessment of textual exercises as helpful, there were downsides like general or short feedback, as well as long correction periods. Participants agree that the assessment process can be accelerated by applying our approach. All of our participants considered structured feedback to be an improvement for the students' comprehension, eight participants agree strongly. The responses support the third hypothesis (H3).

The topic modeling algorithm found 14 topic shifts in our sample of five submissions. The participants derived 15 topic shifts. As visible in Figure 6, 13 topic shifts (92%) are equally detected by the

S1

Differences: [S]
Antipatterns: [S]
-Have one problem [8] and two solutions [8] (one problematic [8] and one refactored) [1-4, 7-10, S, T]
-Antipatterns are a sign of bad architecture [8] and bad coding [1-10, S, T]
Pattern: [S]
-Have one problem and one solution [1-5, 7, 9, 10, S, T]
-Patterns are a sign of elaborated architecutre and coding

S2

The main difference between patterns and antipatterns is, [8] that [6, 7] patterns show you a good way to do something [8, 7] and antipatterns show a bad way to do something. [1, 2, 4-10, S] Nevertheless [7] patterns may become antipatterns in the course of changing understanding of how good software engineering looks like. [1, 2, 5-10, S, T] One example for that is functional decomposition, [5] which used to be a pattern and "good practice". [1, 2, 5, 8, S, T] Over the time it turned out that it is not a goog way to solve problems, so it became a antipattern. [1-10, S, T]

A pattern itsself is a proposed solution to a problem that occurs often and in different situations. [1-3, 5-10, S, T]
In contrast to that a antipattern shows commonly made mistakes when dealing with a certain problem. [2, 7-9, S, T] Nevertheless a refactored solution is aswell proposed.

S3

1.Patterns can evolve into Antipatterns when change occurs [1-8, 10, S, T]
2. [S] Pattern has one solution, [2, 5-8, 10, T] whereas anti pattern can have subtypes of solution [1, 3, 4, 6, 8, 10, S, T]
3. [S] Antipattern has negative consequences [8] and symptom, [2, 6-8, 10] where as patterns looks only into benefits [8] and consequences

S4

Patterns: A way to Model code in differents ways [1-10, S, T]
Antipattern: A way of how Not to Model code

S5

Antipatterns are used when there are common mistakes in software management [5] and development to find these, [1-10, T] while patterns by themselves are used to build software systems [8] in the context of frequent change [8] by reducing complexity and isolating the change. [1-10, S, T]
Another difference is that the antipatterns have problematic solution [5, 8] and then refactored solution, [2, 5, 6, 8-10] while patterns only have a solution.

Figure 6: Submissions S1-S5 from our PSE data set. The submissions were segmented by two algorithms, as well as ten participants. The detected segment borders are marked inline with the text in square brackets: Topic Modeling Algorithm [T], Syntactical Separator Approach [S], and Participants [1-10].

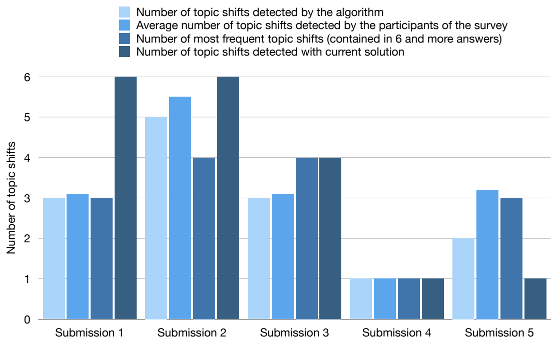


Figure 7: Comparison of the number of detected topic shifts by the current and proposed systems as well as the participants.

newly proposed algorithm and a majority of the participants. Only two topic shifts are not detected by the algorithm (false negative), and one topic shifts detected by the algorithm in S2 has no majority with the participants (false positive). This analysis does support the fourth hypothesis (H4).

5.4 Discussion

We could test the performance of the proposed system and compare it to the current solution based on the topic shifts' positions marked by students. Two interesting details could be discovered.

First, submission S2 was the only case where the proposed solution performed slightly worse than the existing approach. We can explain this with the character of this submission. The student submitted a rather long answer. It consists of seven sentences whereas the average number in our dataset is 3.6. In general, submissions with a lot of sentences where the same information is repeated multiple times can become a challenge. The student also gives an example of an anti-pattern. Answers with examples can become a problem for the proposed solution since there is an unbound set of examples that can be provided and thus it is difficult to judge if the keyword approach suits this case. A solution to this could be dynamically determining the number of keywords.

Second, when reviewing the students' segmentation, there were several answer sheets with significantly more topic shifts than found in other responses. This is usually because the participant saw an "and" in a sentence and decided that there are two different objects or verbs, hence, two different statements. One such case was the following part of an answer: "Antipatterns are used when there are common mistakes in software management and development to find these". Some participants put a boundary between the words "management" and "and". However, this kind of segmentation can lead to problems for further processing and assigning feedback to the text blocks. Though this part of the sentence does have two objects and they could, for example, be correct and incorrect or the other way around, the two resulting text blocks are both incomplete. The first text block misses the "find these" part, the second one — the subject of the sentence. This proves that it is possible to get text blocks that do not make any sense without context. A possible solution could be augmenting the parts of the sentence with the subject or the object from the other part. This, however, demands a deeper analysis of the sentence structure.

During the evaluation, we could make some interesting observations. There are two different types of text blocks that could be treated in another way. First, phrases that express the student's personal opinion about the question or the lecture, like "I do not understand this" or "oh, that's easy", do not need to be assessed. A possibility could be to discover them and exclude them from the corpus to improve the quality of the data for further processing. Incomplete sentences and clauses can also be treated differently. Compound sentences with several clauses often contain multiple different statements. Currently, we do not want to split them up. A sentence like "I like apples and bananas" does have two objects, but a text block "and bananas" does not make any sense without context, the subject and the verb in this case. So a possible solution could be augmenting incomplete text blocks with the corresponding missing context. This could be addressed by implementing PoS tagging.

5.5 Threats to Validity

One of the problems of the evaluation is the small size of the population. The validity could be improved by either increasing the population to include more tutors with different experience levels or by choosing a more experienced population of instructors. In addition, selected submissions for the segmentation task are a threat to external validity since they are from a single lecture. Third, submissions are chosen according to the formatting of the answer, as we allowed different answering formats such as bullet points or full sentences. The study therefore only provides anecdotal evidence on the performance of the assessment algorithm.

6 SUMMARY

In this paper, we have formalized a new algorithm based on topic modeling and text segmentation to segment student answers into topically coherent text blocks. A prototypical implementation has been integrated as part of the open-source Athene project[6] into the automatic assessment management system Artemis. A performance evaluation with ten students has shown that the new algorithm performs better than an algorithm using syntactical separators such as delimiters.

6.1 Conclusion

The presented algorithm is a small building block towards a semi-automated assessment support system for textual exercises, as well as the vision of fully automated assessments of textual exercises. Producing coherent text blocks from student submissions improves the experience for instructors, tutors, and students:

For instructors, a structured form of feedback makes it easier to compare against grading criteria. The increasing degree of automation reduces the workload necessary to conduct textual exercises.

For tutors, the algorithm allows to automate the first step of the grading process and removes some of the overhead related to the segment-based assessment concept. Generated feedback suggestions improve the value of each feedback element, as it can be easily

[6] "Athene: A library to support (semi-)automated assessment of textual exercises," https://github.com/ls1intum/Athene, 2020.

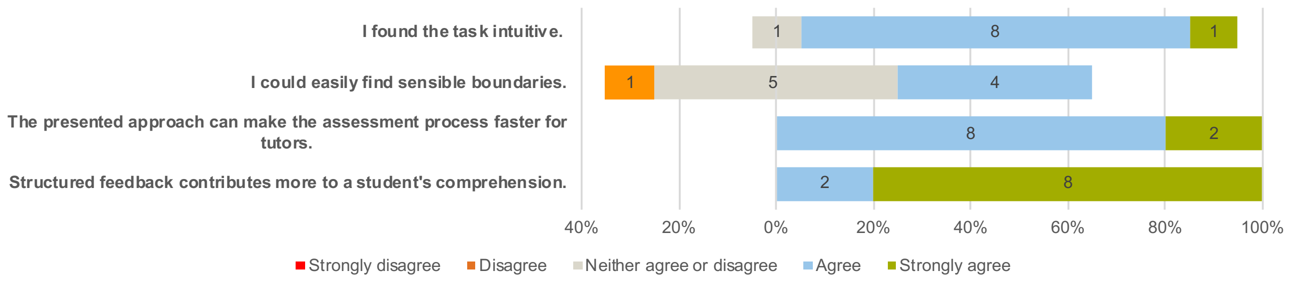


Figure 8: Participants response on their subjective impression of the approach ranked on a five-point scale based on Likert [27]. ($n = 10$)

reused for multiple students, even by other tutors. Suggestions reduce the workload, as a partial assessment is already pre-filled. A semi-automated system should encourage tutors to create extensive and high-quality explanations.

For students, feedback will be more concise. A direct link between a segment of their submission and feedback helps students to understand the feedback and their mistakes. They profit from improvements for tutors, which we envision to lead to quicker and more extensive feedback.

6.2 Future Work

The result of the algorithm's application can be improved in two areas: keywords and text blocks using statistical models, topic models, or decision trees. Additionally, a thesaurus could be used to recognize synonyms.

The effect of the algorithm on the assessment system can be evaluated in two aspects: The usability for tutors when grading text blocks and the impact of the segmentation on the quality of feedback suggestions.

REFERENCES

[1] Doug Beeferman, Adam L. Berger, and John D. Lafferty. 1997. Text Segmentation Using Exponential Models. *CoRR* (1997). http://arxiv.org/abs/cmp-lg/9706016

[2] Jan Philip Bernius and Bernd Bruegge. 2019. Toward the Automatic Assessment of Text Exercises. In *2nd Workshop on Innovative Software Engineering Education (ISEE)*. Stuttgart, Germany, 19–22.

[3] Jan Philip Bernius, Anna Kovaleva, and Bernd Bruegge. 2020. Segmenting Student Answers to Textual Exercises Based on Topic Modeling. In *17th Workshop Software Engineering im Unterricht der Hochschulen (SEUH)*. Innsbruck, Austria, 72–73.

[4] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python* (1st ed.). O'Reilly Media, Inc.

[5] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *The Journal of Machine Learning Research* 3 (2003), 993–1022.

[6] Charles C. Bonwell and James A. Eison. 1991. *Active Learning: Creating Excitement in the Classroom.* ERIC Clearinghouse on Higher Education.

[7] Qiuxing Chen, Lixiu Yao, and Jie Yang. 2016. Short text classification based on LDA topic model. In *2016 International Conference on Audio, Language and Image Processing (ICALIP)*. IEEE, 749–753. https://doi.org/10.1109/icalip.2016.7846525

[8] Freddy Y. Y. Choi. 2000. Advances in Domain Independent Linear Text Segmentation. In *Proceedings of the 1st North American Chapter of the Association for Computational Linguistics Conference* (Seattle, Washington) *(NAACL 2000)*. Association for Computational Linguistics, USA, 26–33.

[9] Jacob Eisenstein. 2009. Hierarchical Text Segmentation from Multi-Scale Lexical Cohesion. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, Boulder, Colorado, 353–361. https://www.aclweb.org/anthology/N09-1040

[10] Jacob Eisenstein and Regina Barzilay. 2008. Bayesian Unsupervised Topic Segmentation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (Honolulu, Hawaii) *(EMNLP '08)*. Association for Computational Linguistics, USA, 334–343.

[11] David W. Embley, Douglas M. Campbell, Randy D. Smith, and Stephen W. Liddle. 1998. Ontology-based Extraction and Structuring of Information from Data-rich Unstructured Documents. In *Proceedings of the seventh international conference on Information and knowledge management - CIKM '98*. ACM Press, 52–59. https://doi.org/10.1145/288627.288641

[12] Pavlina Fragkou, Vassilios Petridis, and Athanasios Kehagias. 2004. A Dynamic Programming Algorithm for Linear Text Segmentation. *Journal of Intelligent Information Systems* 23, 2 (2004), 179–197. https://doi.org/10.1023/b:jiis.0000039534.65423.00

[13] Arthur C. Graesser, Peter Wiemer-Hastings, Katja Wiemer-Hastings, Derek Harter, Tutoring Research Group Tutoring Research Group, and Natalie Person. 2000. Using Latent Semantic Analysis to Evaluate the Contributions of Students in AutoTutor. *Interactive Learning Environments* 8, 2 (2000), 129–147. https://doi.org/10.1076/1049-4820(200008)8:2;1-b;ft129

[14] Michael A. K. Halliday and Ruqaiya Hasan. 1976. *Cohesion in English.* Longman, London.

[15] Marti A. Hearst. 1997. TextTiling: Segmenting Text into Multi-Paragraph Subtopic Passages. *Computational Linguistics* 23, 1 (1997), 33–64.

[16] Anette Hulth. 2003. Improved Automatic Keyword Extraction Given More Linguistic Knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing -*. Association for Computational Linguistics, 216–223. https://doi.org/10.3115/1119355.1119383

[17] Anil K. Jain and Sushil Bhattacharjee. 1992. Text segmentation using gabor filters for automatic document processing. *Machine Vision and Applications* 5, 3 (1992), 169–184. https://doi.org/10.1007/bf02626996

[18] Tibor Kiss and Jan Strunk. 2006. Unsupervised Multilingual Sentence Boundary Detection. *Computational Linguistics* 32, 4 (2006), 485–525. https://doi.org/10.1162/coli.2006.32.4.485

[19] Jan Knobloch and Enrico Gigantiello. 2017. AMATI: Another Massive Audience Teaching Instrument. In *15th Workshop Software Engineering im Unterricht der Hochschulen (SEUH)*. Hannover, Germany, 63–68.

[20] Takafumi Koshinaka, Ken ichi Iso, and Akitoshi Okumura. 2005. An HMM-based text segmentation method using variational Bayes approach and its application to LVCSR for broadcast news. In *Proceedings. (ICASSP '05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, Vol. 1. IEEE, 485–488. https://doi.org/10.1109/icassp.2005.1415156

[21] Omri Koshorek, Adir Cohen, Noam Mor, Michael Rotman, and Jonathan Berant. 2018. Text Segmentation as a Supervised Learning Task. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, Vol. 2. Association for Computational Linguistics, 469–473. https://doi.org/10.18653/v1/n18-2075

[22] Stephan Krusche and Andreas Seitz. 2018. ArTEMiS: An Automatic Assessment Management System for Interactive Learning. In *49th ACM Technical Symposium on Computer Science Education.* ACM, 284–289. https://doi.org/10.1145/3159450.3159602

[23] Stephan Krusche and Andreas Seitz. 2019. Increasing the Interactivity in Software Engineering MOOCs - A Case Study. In *31th Conference on Software Engineering Education and Training (CSEE&T)*.

[24] Stephan Krusche, Andreas Seitz, Jürgen Börstler, and Bernd Bruegge. 2017. Interactive Learning: Increasing Student Participation Through Shorter Exercise Cycles. In *19th Australasian Computing Education Conference.* ACM, 17–26. https://doi.org/10.1145/3013499.3013513

[25] Stephan Krusche, Nadine von Frankenberg, Lara Marie Reimer, and Bernd Bruegge. 2020. An Interactive Learning Method to Engage Students in Modeling. In *Proceedings of the 42nd International Conference on Software Engineering - Software Engineering Education and Training (ICSE-SEET'20)*. Seoul, South Korea.

[26] Rainer Lienhart and Wolfgang Effelsberg. 2000. Automatic text segmentation and text recognition for video indexing. *Multimedia Systems* 8, 1 (2000), 69–81. https://doi.org/10.1007/s005300050006

[27] Rensis Likert. 1932. A Technique for the Measurement of Attitudes. *Archives of Psychology* 22, 140 (1932), 1–55.

[28] Yutaka Matsuo and Mitsuru Ishizuka. 2003. Keyword Extraction from a Single Document using Word Co-occurrence Statistical Information. *International Journal on Artificial Intelligence Tools* 13, 01 (2003), 157–169. https://doi.org/10.1142/s0218213004001466

[29] Richard E. Mayer, Andrew Stull, Krista DeLeeuw, Kevin Almeroth, Bruce Bimber, Dorothy Chun, Monica Bulger, Julie Campbell, Allan Knight, and Hangjin Zhang. 2009. Clickers in college classrooms: Fostering learning with questioning methods in large lecture classes. *Contemporary Educational Psychology* 34, 1 (2009), 51–57. https://doi.org/10.1016/j.cedpsych.2008.04.002

[30] Olena Medelyan and Ian H. Witten. 2006. Thesaurus Based Automatic Keyphrase Indexing. In *Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries* (Chapel Hill, NC, USA) *(JCDL '06)*. Association for Computing Machinery, New York, NY, USA, 296–297. https://doi.org/10.1145/1141753.1141819

[31] George A. Miller. 1995. WordNet: A Lexical Database for English. *Commun. ACM* 38, 11 (1995), 39–41. https://doi.org/10.1145/219717.219748

[32] Hemant Misra, François Yvon, Olivier Cappé, and Joemon Jose. 2011. Text segmentation: A topic modeling perspective. *Information Processing & Management* 47, 4 (2011), 528–544. https://doi.org/10.1016/j.ipm.2010.11.008

[33] Hemant Misra, François Yvon, Joemon M. Jose, and Olivier Cappe. 2009. Text Segmentation via Topic Modeling: An Analytical Study. In *Proceeding of the 18th ACM conference on Information and knowledge management - CIKM '09* (Hong Kong, China). ACM Press, 1553–1556. https://doi.org/10.1145/1645953.1646170

[34] Irina Pak and Phoey Lee Teh. 2017. Text Segmentation Techniques: A Critical Review. In *Innovative Computing, Optimization and Its Applications: Modelling and Simulations*. Springer International Publishing, 167–181. https://doi.org/10.1007/978-3-319-66984-7_10

[35] Jay M. Ponte and W. Bruce Croft. 1997. Text segmentation by topic. In *Research and Advanced Technology for Digital Libraries*. Springer Berlin Heidelberg, 113–125.

[36] Martin F. Porter. 1980. An algorithm for suffix stripping. *Program: electronic library and information systems* 14, 3 (1980), 130–137.

[37] Ann Poulos and Mary Jane Mahony. 2008. Effectiveness of feedback: the students' perspective. *Assessment & Evaluation in Higher Education* 33, 2 (2008), 143–154. https://doi.org/10.1080/02602930601127869

[38] Juan Enrique Ramos. 2003. Using TF-IDF to Determine Word Relevance in Document Queries. In *1st instructional Conference on Machine Learning*.

[39] Jeffrey C. Reynar. 1999. Statistical Models for Topic Segmentation. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics -*. Association for Computational Linguistics, 357–364. https://doi.org/10.3115/1034678.1034735

[40] Martin Riedl and Chris Biemann. 2012. Sweeping through the Topic Space: Bad Luck? Roll Again!. In *Proceedings of the Joint Workshop on Unsupervised and Semi-Supervised Learning in NLP* (Avignon, France) *(ROBUS-UNSUP '12)*. Association for Computational Linguistics, USA, 19–27.

[41] Martin Riedl and Chris Biemann. 2012. TopicTiling: A Text Segmentation Algorithm Based on LDA. In *Proceedings of ACL 2012 Student Research Workshop* (Jeju Island, Korea) *(ACL '12)*. Association for Computational Linguistics, USA, 37–42.

[42] C. Osvaldo Rodriguez. 2012. MOOCs and the AI-Stanford like Courses: Two Successful and Distinct Course Formats for Massive Open Online Courses. *European Journal of Open, Distance and E-Learning* (2012).

[43] Yuwei Tu, Ying Xiong, Weiyu Chen, and Christopher Brinton. 2018. A Domain-Independent Text Segmentation Method for Educational Course Content. *IEEE International Conference on Data Mining Workshops* (2018). https://doi.org/10.1109/icdmw.2018.00053

[44] Yaakov Yaari. 1997. Segmentation of Expository Texts by Hierarchical Agglomerative Clustering. *CoRR* (1997). arXiv:9709015 [cmp-lg]

[45] Jon P. Yamron, Ira Carp, Larry Gillick, Steve Lowe, and Paul van Mulbregt. 1998. A hidden Markov model approach to text segmentation and event tracking. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP '98 (Cat. No.98CH36181)*, Vol. 1. IEEE, IEEE, 333–336. https://doi.org/10.1109/icassp.1998.674435

7 A Machine Learning Approach for Suggesting Feedback in Textual Exercises in Large Courses

Publication [BKB21] has been published as a **peer-reviewed conference paper**:

Authors:	Jan Philip Bernius, Stephan Krusche, and Bernd Bruegge
Conference:	8th ACM Conference on Learning @ Scale (L@S '21)
Location:	Potsdam, Germany
Pages:	173–182
Year:	2021
DOI:	10.1145/3430895.3460135
Review:	Peer Reviewed (4 Reviewers)
Acc. Rate:	30%

Summary This paper formalizes CoFee, a machine learning approach designed to suggest Computer-aided Feedback for textual exercises. CoFee applies clustering to group the text segments by similarity so that the same feedback can be applied to all segments within the same cluster.

Athena[1] is the reference implementation of the CoFee design. Athena offers a service for learning management systems to identify clusters of similar text segments within student submissions.

An empirical evaluation of Athena reviewed 17 textual exercises in two large courses at the Technical University of Munich with 2,300 registered students and 53 instructors. Athena suggested feedback for 26% of the submissions. The instruc-

[1]In the publication [BKB21] we refer to Athena as Athene. Both names can be considered synonyms.

tors accepted 85% of these suggestions, and 5% were extended with an instructor comment before releasing them to the students.

Contributions **J.P. B.** conceptualized and formalized the CoFee framework, as well as the design, architecture and implementation of Athena. **J.P. B.** and S.K. developed the study design, **J.P. B.** analyzed the data and evaluated the results. **J.P. B.** wrote and visualized the paper. B. B. provided feedback and helped improving the manuscript. S. K. and B. B. reviewed the paper.

Conference paper The author's version of the conference paper is reprinted in this dissertation. The final version of record is available at:
`https://dl.acm.org/doi/10.1145/3430895.3460135`

A Machine Learning Approach for Suggesting Feedback in Textual Exercises in Large Courses

Jan Philip Bernius, Stephan Krusche, and Bernd Bruegge
Department of Informatics
Technical University of Munich
Munich, Germany
janphilip.bernius@tum.de, krusche@in.tum.de, bernd.bruegge@tum.de

ABSTRACT

Open-ended textual exercises facilitate the comprehension of problem-solving skills. Students can learn from their mistakes when teachers provide individual feedback. However, courses with hundreds of students cause a heavy workload for teachers: providing individual feedback is mostly a manual, repetitive, and time-consuming activity.

This paper presents **CoFee**, a machine learning approach designed to suggest computer-aided feedback in open-ended textual exercises. The approach uses topic modeling to split student answers into text segments and language embeddings to transform these segments. It then applies clustering to group the text segments by similarity so that the same feedback can be applied to all segments within the same cluster.

We implemented this approach in a reference implementation called **Athene** and integrated it into Artemis. We used Athene to review 17 textual exercises in two large courses at the Technical University of Munich with 2,300 registered students and 53 teachers. On average, Athene suggested feedback for 26% of the submissions. Accordingly, 85% of these suggestions were accepted by the teachers, 5% were extended with a comment and then accepted, and 10% were changed.

Author Keywords
Software Engineering, Education, Interactive Learning, Automatic Assessment, Grading, Assessment Support System, Learning, Feedback

CCS Concepts
•**Social and professional topics** → **Software engineering education;** •**Computing methodologies** → *Natural language processing;*

INTRODUCTION

The rise in student numbers in universities has led to an increase in course management efforts, and made it challenging to provide high-quality individual feedback to students [19]. Recent approaches, such as online platforms and live streaming, allow teachers[1] to cope and interact with a large amount of students on an individual level, regardless of the respective course size.

In particular, large university courses with hundreds of students rely on teaching assistants to provide feedback on exercises, e.g., multiple-choice quizzes and textual exercises. Multiple-choice quizzes are easy to assess, and tools are broadly available in learning management systems (LMSs) and for paper-based assessment. However, mastery of these quizzes does not require problem-solving skills because they typically target only lower cognitive skills, in particular, *knowledge* recall and *comprehension*. Most quiz types include predefined options and do not reflect work practices in industry. It is difficult to create quizzes that stimulate higher cognitive skills, such as problem-solving, which are important in computer science [1, 32].

Open-ended textual exercises allow 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 which can be characterized by words and phrases. The searchlight theory of scientific knowledge [27] states that students increase their knowledge through observations, especially observations that prove their assumptions wrong. Students profit from having an individual feedback relationship with their teachers [10]. Individual feedback and formative assessments are essential elements in learning [11, 12]. Feedback on open-ended exercises allows students to try out problem-solving and to experience failure. Students need guidance in the form of feedback in their learning activities to prevent misconceptions [13].

[1]For this paper, we define teachers as both instructors and teaching assistants (see Figure 1). Instructors are employees of the university such as professors, lecturers, and doctoral candidates. Teaching assistants are experienced students who have passed the same course previously with a good grade and who are motivated to help in the teaching process. Some universities also use the term "tutor" to refer to a teaching assistant.

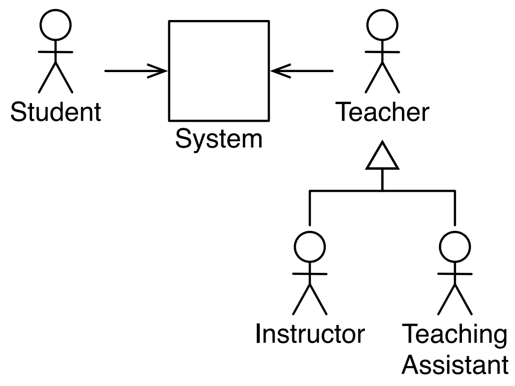


Figure 1. Use case diagram of the Athene and Artemis system. Students and teachers interact with the system. Teachers are instructors, employees of the university, or teaching assistants, who are previous students hired to assist in teaching.

However, textual exercises lead to greater variability because students need to formulate individual answers to problems. This results in high manual effort when reviewing students' answers. Assuring consistent feedback is difficult with a large number of teaching assistants. In this paper, we present an approach for computer-aided feedback for textual exercises that addresses these challenges. We implemented the approach in an open-source reference implementation, used it in multiple exercises, and evaluated its effects on the learning experience of students. In particular, we investigated the following research questions (RQ):

RQ1 **Coverage:** How much feedback can be automatically suggested?

RQ2 **Accuracy:** How accurate is the suggested feedback?

RQ3 **Quality:** How do students perceive the quality of the automatically suggested feedback?

The paper is organized as follows: Section 2 describes the background of this work which consists of machine learning concepts, in particular language models. In Section 3, we show similar approaches and relate them to the approach presented in this paper. Section 4 presents the approach computer-aided feedback for textual exercises (CoFee) based on supervised learning to deal with the greater variability in the student answers. Language embeddings and clustering are used to provide individual feedback based on similarity. Section 5 shows the open-source reference implementation **Athene**[2] which is integrated into the open-source LMS Artemis[3]. Section 6 describes the courses in which the approach was used, shows the study design of the empirical evaluation with respect to Bloom's revised taxonomy [2], presents results and limitations, and discusses the findings. Section 7 concludes the paper with its main contributions and future work.

BACKGROUND: LANGUAGE MODELS

Assessing text submissions automatically requires comparing segments of those submissions and identifying similar pieces of text. Therefore, we need a measurable abstraction of a texts meaning as an intermediate representation. This paper relies on existing approaches and techniques from the domain of natural language processing (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 use of 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) [21]. The feature vector represents different aspects of the word and consequently, words that have the same meaning are assigned similar vector representations. Additionally, word embeddings are capable of capturing word analogies by examining various dimensions of the differences between word vectors [24]. 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 words. This allows words that are used in similar contexts to have similar representations, naturally capturing their meaning. ELMo [26] is a word embedding constructed as a task-specific combination of the intermediate layer representations in a bidirectional language model (biLM). 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 (LM), the higher-level long short term memory (LSTM) 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 LM, ELMo is able to capture both characteristics of the language. ELMo representations have three main characteristics that allow them 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.

RELATED WORK

Automated essay scoring (AES) computes scores on written solutions based on previous submissions. AES systems require a perfect solution to be available up front [23, 31]. They primarily consider the distance to a perfect solution to determine the grade. Feedback is not the focus. Manual clustering and shared grading are concepts used in research [25] and commercial tools (i.e., Gradescope). Managing clusters is hard at scale, communicating the exact differences between clusters between many graders.

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

Atenea is a computer-assisted assessment system for scoring short answers in computer science [25] and is integrated into a web-based application. Atenea uses a database of questions with a correct sample solution for each, either written by a teacher or taken from a highly graded student's answer. When a student accesses Atenea, a random question from this pool is asked and compared to the given sample solution by utilizing a hybrid for syntax as well as semantic similarity. The system works by combining latent semantic analysis (LSA) and a modified bilingual evaluation understudy (BLEU) algorithm, with the hypothesis that syntax and semantics complement each other naturally. The combination of both NLP tools always performs better (with a higher hit rate) than their individual parts, with the authors believing that combinations of syntactical and semantical analysis can lead to even greater results for automatic text assessment.

Atenea compares student answers to a set of predefined answers. The grade is determined by its similarity to these predefined answers. This approach is limited to exercises with a narrow answer space where possible answers are known beforehand. A high variability in answers requires a large set of predefined answers, which limits the applicability of the system. The focus of the Atenea system is grading, whereas Athene is primarily focused on individual feedback. Athene 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 Athene by using it in multiple courses and measuring its performance.

Powergrading is an automatic assessment approach [3]. Instead of solely focusing on providing a numerical score or a right or wrong grade, Powergrading tries to justify a certain given grade by providing feedback in the form of a comment as to why an answer is right or wrong, similar to how a teacher would do it in a classroom setting. Basu et al. propose a system, that clusters similar answers to a question so that teachers can "divide and conquer" the correction process by assigning 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, which is composed of different features and tries to learn a similarity metric between two students' answers automatically. 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)[4] 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 and concluded that around 97% of all submissions can be grouped into similar clusters so that teachers 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 [3].

[4]TF-IDF: An information extraction statistic which indicates how significant a word is to a document [28].

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

Gradescope is a system geared toward the assessment of handwritten homework and exam exercises [30] by scanning paper-based work. Teachers review the submissions online. Gradescope allows the teacher to dynamically create grading rubrics at the assessment time. For the assessment, teachers can group similar submissions manually for shared grading or relay on suggested groups.[5]

Athene follows a similar idea by sharing feedback with groups of answers; however, Athene groups individual segments, whereas Gradescope groups entire submissions. Gradescope allows the grader to grade multiple submissions as one, similar to Powergrading, whereas Athene shares individual feedback elements across multiple submissions. Athene requires teachers to inspect every submission and supports 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, for exercises with high variability in answers (e.g., when asking for examples), this approach is more limited as more groups with less elements need to be reviewed.

APPROACH: COMPUTER-AIDED FEEDBACK (COFEE)

CoFee uses supervised machine learning to learn correct answers and related feedback. Figure 2 shows the main workflow how CoFee can automatically propose computer-aided feedback to students' answers. CoFee learns which answers to an exercise are correct and which are incorrect. For further submissions, the learning platform can automatically generate suggestions for similar answers or even evaluate the answers fully automatically. In doing so, the learning platform uses the knowledge of previous assessments by lecturers. The more students participate in an exercise, the more knowledge is generated and the better feedback the learning platform can suggest.

Figure 3 shows the details of the activity "preprocess answers" shown in Figure 2 and represents the basis for the three objectives mentioned above. The system analyzes incoming text (responses) using NLP, divides them into text segments, and uses them to create text clusters with similar text segments from different responses. This is done using a combination of segmentations and linguistic embeddings, in particular deeply contextualized word representations (ELMo). This allows for an understanding students' responses and for the generation of individualized feedback. In this way, a learning platform can automatically reuse manual feedback for contributions

[5]https://gradescope.com

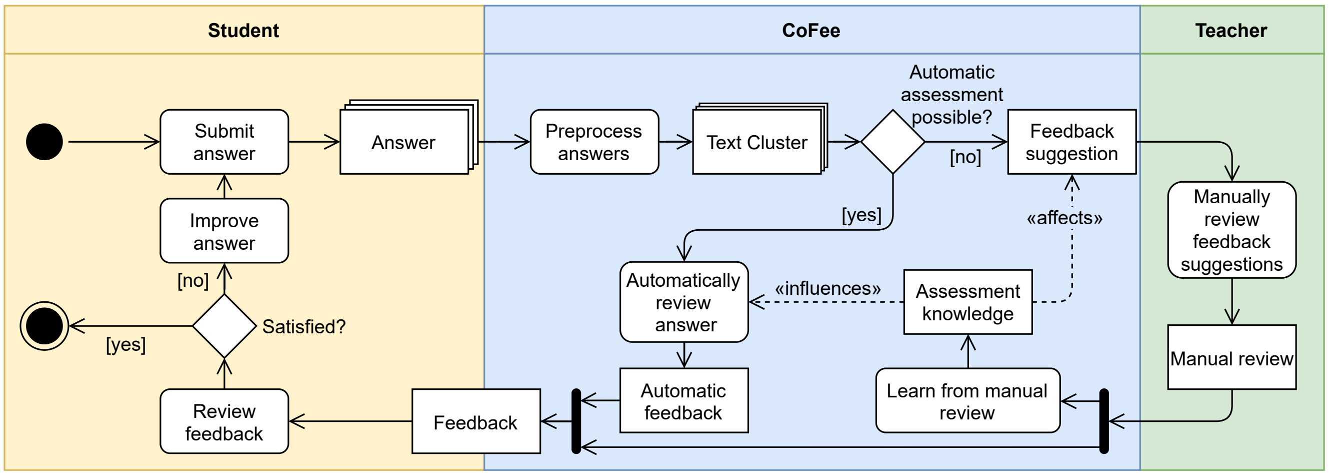


Figure 2. Workflow of automatic assessment of submissions to textual exercises based on the manual feedback of teachers. CoFee analyzes manual assessments and generates knowledge for the suggestion of computer-aided (automatic) feedback (UML activity diagram).

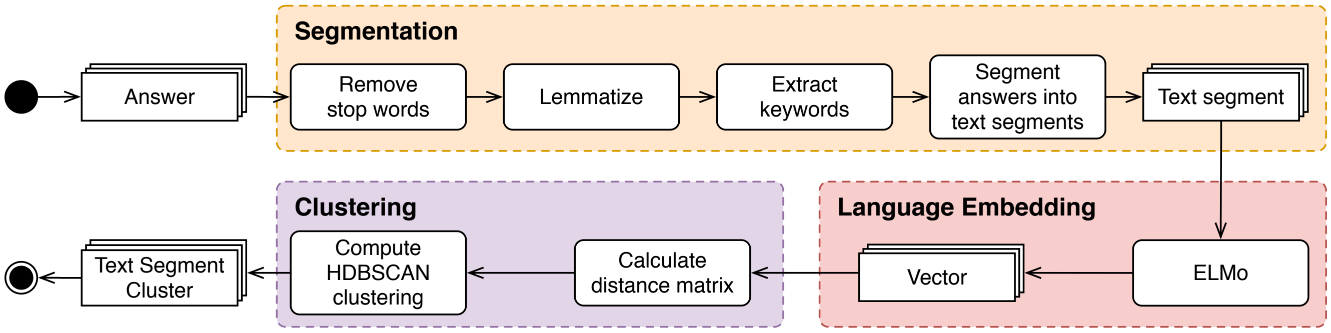


Figure 3. Detailed overview of the machine learning activities as part of the "preprocess answers" activity in Figure 2. These are used to extract text segments and build text clusters for scoring and similarity analysis (UML activity diagram).

from different students. This can reduce the workload for teachers and increase the consistency and quality of feedback to improve students' understanding.

The goal is to increase the quality and quantity of the feedback provided to students while decreasing the overall assessment time. 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 [5], requiring assessors to provide feedback and score in relation to a segment of student's answer, resulting in relatable and reusable feedback elements.

CoFee trains its assessment model with every feedback element and thereby 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 distance between two text segments should be proportional to the distance between the feedback comments. If this relation is violated, CoFee prompts the teacher to 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 teachers have the chance to resolve inconsistencies.

Compared to existing work, our system segments and clusters student solutions automatically. By training the system during the assessment process, we do away with the need for a reference dataset before the assessment. Furthermore, by training with highly and lowly scored solutions, we maintain a dataset to provide helpful feedback comments to support the learning process. Dynamically collecting the dataset during assessment keeps the system independent of any domain and allows for use of the system with new exercises to incorporate the latest knowledge into teaching.

REFERENCE IMPLEMENTATION (ATHENE)

We implemented CoFee in a reference implementation called Athene [4] that is integrated into the learning platform Artemis [15]. After the exercise deadline, Artemis sends the students' answers to Athene for processing. Athene will preprocess the answers before the assessment begins and will identify segments suitable for the same feedback. Figure 3 depicts the preprocessing activities. This represents the basis for the three objectives mentioned above. 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 responses. Figure 4 depicts the top-level design of the system which consists of three steps: segmentation, language embedding and clustering.

First, Athene analyzes the answers to identify segments [6, 7]. Therefore, Athene identifies common topics described in the answers from all students. A topic is represented by a keyword. To identify the important topics for an exercise, Athene counts the occurrences of lemmatized words across all students and selects the 10 most common words [7]. Within each student answer, Athene will break down all submissions into clauses. Adjacent clauses that share the same topic, represented by the use of a keyword and absence of a new keyword, are merged to form a segment. If a new keyword appears an a following clause, we identify a topic shift and start a new segment. This results in a set of topically coherent segments.

Second, Athene 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 for the identification of similarities. Athene uses a pre-trained ELMo model [26] based on a dataset consisting of 5.5B tokens from Wikipedia and news articles.[6]

Third, Athene employs the Hierarchical Density-Based Spatial Clustering (HDBSCAN) clustering algorithm [22] to identify classes of similar text segments. Within a cluster, Athene shares manually created feedback as suggestions. The hierarchical clustering algorithm allows for a determination of the required number of clusters dynamically. Further, the hierarchical structure is used to dynamically narrow or widen the search radius depending on the availability of feedback. Narrow clusters provide more accurate feedback on the one side; however, they also limit the possible coverage. Larger clusters increase the possibility to find existing feedback to compose a suggestion; however, they also increase the risk of false feedback.

During the manual assessment, Athene sorts submissions so that it priorities submissions with the highest effect on automated grading. Submissions with several segments in clusters without feedback are prioritized, maximizing the possible coverage for automatic feedback suggestions. For each segment, Athene searches their respective clusters for existing feedback and suggests the closest feedback. Furthermore, credit points associated with feedback are used to prioritize based on the clusters' credit average. Athene's automatic feedback suggestions are displayed to teachers within Artemis as part of the review interface [5], as depicted in Figure 5. Teachers can add additional feedback to unreviewed parts of the student solution. They can either approve of the feedback suggestions or update them as they see fit.

EVALUATION

After several teachers used Athene in initial experiments in smaller courses with around 500 students, they found anecdotal evidence that the system improves the quantity and quality of feedback. The next step was to evaluate the approach in multiple exercises in the course *Introduction to Software Engineering (SE1)* with 1,800 students and 49 teaching assistants and in a second course *Networks for Monetary Transactions*. In this section, we describe the two courses and the study

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

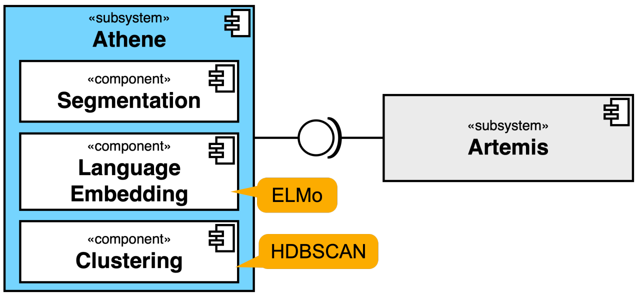
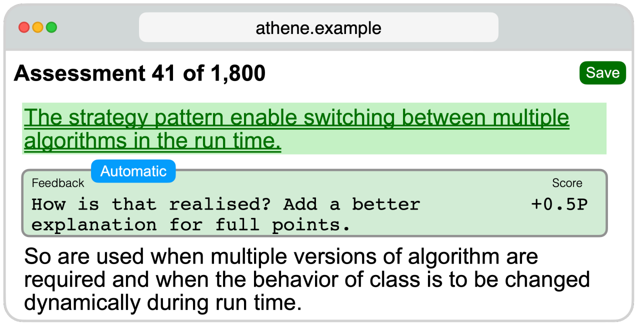


Figure 4. Top-level design of Athene, which is decomposed into three subsystems for segmentation, language embedding, and clustering and offers an API to be used in existing LMS (UML component diagram).



Assessment 41 of 1,800

The strategy pattern enable switching between multiple algorithms in the run time.

```
How is that realised? Add a better      +0.5P
explanation for full points.
```

So are used when multiple versions of algorithm are required and when the behavior of class is to be changed dynamically during run time.

Figure 5. Example of the teacher interface: Athene presents a feedback suggestion for the first text segment with a feedback comment and a score.

design of the evaluation. We show the results of the usage of Athene and discuss the findings, implications, and limitations.

Courses

The course SE1 is an introductory software engineering course, with around 1,800 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, and project management and covers UML modeling [19]. To participate in the course, students need to have fundamental programming experience (e.g., CS1). The instructors use constructive alignment [8] 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 [2]. The course focuses on higher cognitive processes: students apply the concepts in concrete exercises.

Following an interactive learning approach, SE1 teaches software engineering concepts with multiple, small iterations of theory, example, exercise, solution and reflection [16]. It utilizes exercises to foster student participation [17] and to motivate the students to attend the lectures [18]. The course involves different kinds of exercises:

1. Lecture exercises as part of the (virtual) lectures
2. Group exercises solved in small ad hoc groups
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 exercises 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. To train software engineering and problem-solving skills, the instructors utilize programming, modeling, **textual**, and quiz exercises in the course. Automatic assessment suggestions based on Athene have been enabled for 11 textual homework exercises and six textual exam exercises.

The course *Networks for Monetary Transactions* has the learning goals to understand and assess the fundamentals, architecture, and security of domestic and international payment networks and their legal frameworks. Around 500 students participated. The teachers used Artemis to conduct an online exam during the COVID-19 pandemic. The exam consisted of 11 quiz exercises and three text exercises. Automatic assessment suggestions based on Athene were enabled for one **textual** exam exercise: *IT-Attacks*.

Bloom created the taxonomy of educational objectives, defining six categories: Knowledge, Comprehension, Application, Analysis, Synthesis, and Evaluation [9]. The *revised taxonomy* shifts the focus from static educational objectives toward a classification of cognitive processes students encounter when solving exercises [2]. The exercises conducted as part of the evaluation can be classified to train the cognitive processes *Remember*, *Understand*, but *Apply*, and *Analyze* (see Figure 6).

Table 1 lists the textual exercises and includes the cognitive process (as a category) that receives the most training in terms of the revised taxonomy. Some exercises such as *H09E02* and *H10E01* facilitate understanding by asking student to explain concepts. *H10E01*, e.g., states: "Name and explain similarities and differences between the Unified Process and Scrum in your own words". Other exercises such as *Exam 3* focus on the application of knowledge. Students need to apply their requirements elicitation skills in order to create use case descriptions based on a given problem statement.

Study Design

Figure 7 shows the study design of the evaluation that was instantiated for each exercise in which Athene was used for grading. The teacher defines the exercise in Artemis with a problem statement, grading criteria, example solutions, and a due date. The students can insert their solution in plain text on Artemis. After the due date, Artemis sends all student answers to Athene to preprocess the answers as described in the Approach section. The teachers can start reviewing the student answers as soon as Athene completes the preparation and stores the text clusters. For every student answer, the teachers create a review consisting of multiple feedback items. During the review phase, the teachers used a chat room to discuss the grading criteria as needed.

Every review can either be computer-aided, if at least one feedback item is suggested by the system, or manual. Furthermore,

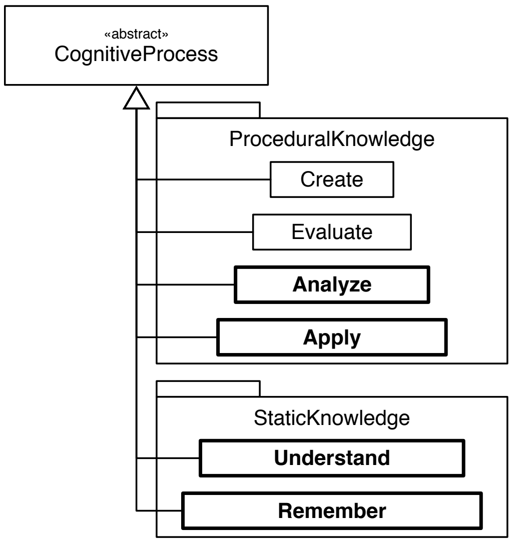


Figure 6. Exercises in the evaluation assess different cognitive processes. This taxonomy, based on the revised Bloom's taxonomy [2], depicts the hierarchy of skills. Exercises test static knowledge by testing the *remember* and *understand* skills but also *apply* and *analyze*, e.g., by identifying design issues from a system.

Athene stores intermediate versions of all feedback items to evaluate how teachers work with feedback suggestions.

After the teachers 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 Athene. 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.

Results

The presentation of the results is based on the research questions stated at the beginning of the paper on the coverage, accuracy and quality of the approach.

Review Coverage

First, we classify the reviews into two categories: manual and computer-aided. A review is considered computer-aided, if at least one feedback item was suggested by Athene. Figure 8 depicts the classification of the reviews. On average, 26% of all reviews were computer-aided. The system performed best in exercise `Exam 1`, with 70% computer-aided reviews. Exercises `Exam 4` and `Exam 6` have the least coverage, with 2% and 8% computer-aided reviews, respectively. However, Athene was disabled for exercise `Exam 4` after a few assessments.

Finding 1: Coverage: Athene can cover up to 70% of reviews with feedback suggestions without previous training data or a predefined solution.

Exercise	Title	Category
H04E01	Coupling and Cohesion	Understand
H04E02	Analysis Models & System Design	Analyze
H04E03	Design Goal Trade-offs	Apply
H05E02	Centralized vs Decentralized Designs	Understand
H06E03	Specification & Implementation Inheritance	Apply
H06E04	Inheritance vs. Delegation	Understand
H07E03	MVC & Observer Pattern	Understand
H09E01	Advantages and Disadvantages of Scrum	Understand
H09E02	Unified Process and Scrum	Understand
H10E01	Problems using Git	Understand
H10E02	Merge Conflicts & Best Practices	Understand
Exam 1	Requirements	Apply
Exam 2	Visionary Scenarios	Apply
Exam 3	Use Cases	Apply
Exam 4	Access Control	Apply
Exam 5	Design Goal Trade-offs	Apply
Exam 6	Centralized vs. Decentralized Control	Apply
Exam 7	IT-Attacks	Remember

Table 1. Homework and exam textual exercises and their categorization following Bloom's revised taxonomy [2] used in the evaluation.

Feedback Accuracy

Second, we classified feedback items based on the intermediate versions collected during the review process. Feedback items can be classified as follows:

1. A feedback suggestion that remains unchanged is classified as *automatic*.
2. For changed suggestions, Athene computes the Levenshtein distance [20] between feedback comments. Athene classifies a changed feedback as a *typo* fix for a Levenshtein distance > 0.9.
3. Athene uses the longest common substring length and the Jaro–Winkler distance [33] to recognize feedback suggestions with a manual *extended* comment.
4. Feedback not classified in these metrics is considered as *changed*.

We analyzed the teachers' assessment work for two homework and seven exam exercises. The results depicted in Figure 9 show that on average, 85% of computer-aided feedback comment suggestions remained unchanged in their final assessment or only had minor modifications, such as corrections to typing mistakes. Furthermore, 5% of suggested comments were extended with additional feedback at the end of the suggestion to provide more details for the student. The remaining 10% of comments were changed. In these cases, the comment was either rewritten from scratch or was heavily revised.

Finding 2: Accuracy: On average, 85% of the feedback suggestions are accurate and can be published to students without modification.

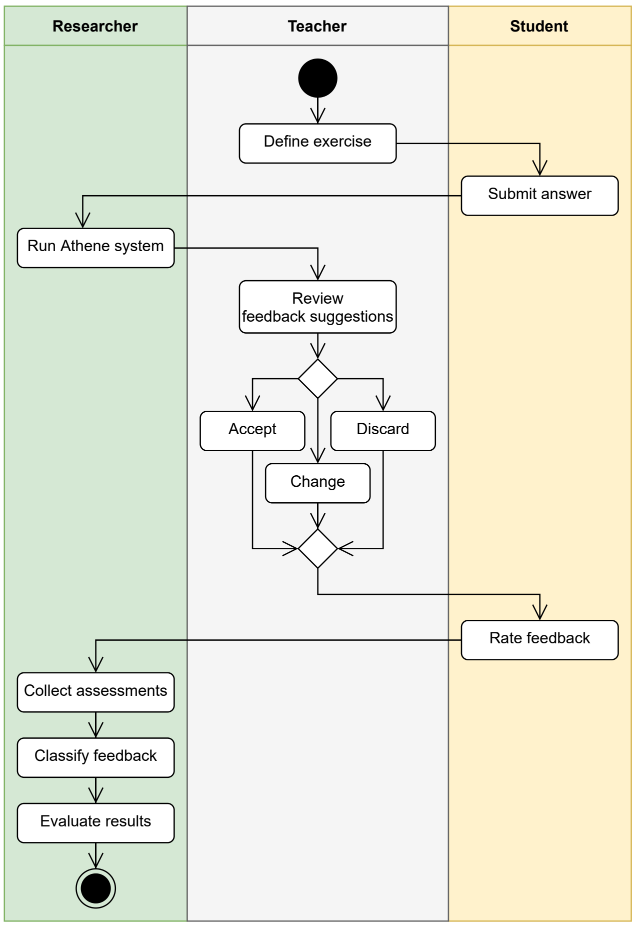


Figure 7. Research approach depicted with the involved actors and flow of events (UML activity diagram).

Perceived Quality

Third, we asked students to rate their received feedback on a 5-star scale. The students rated a total of 396 reviews out of 10,240 total reviews done by the teachers. Artemis presents the rating input underneath the feedback and asks, *"How useful is the feedback for you?"* Figure 10 depicts the distribution by star rating. In the study, 85% of the ratings were either 1-star or 5-star ratings. Students with computer-aided feedback were more likely to give a 5-star rating (72%) when compared to students who received manual feedback (62%). On the same page, computer-aided feedback received 1-star ratings less often (14%) than manual feedback (25%). Students giving a 5-star rating on average (92% and 89%, respectively) had better scores than students giving 1-star ratings (69% and 66%, respectively).

Finding 3: Quality: The computer-aided feedback in Athene has at least the same quality as manual feedback.

Limitations

This section discusses threats to the trustworthiness of the presented results, and whether the results are biased based on the researchers' subjective point of view. We distinguish

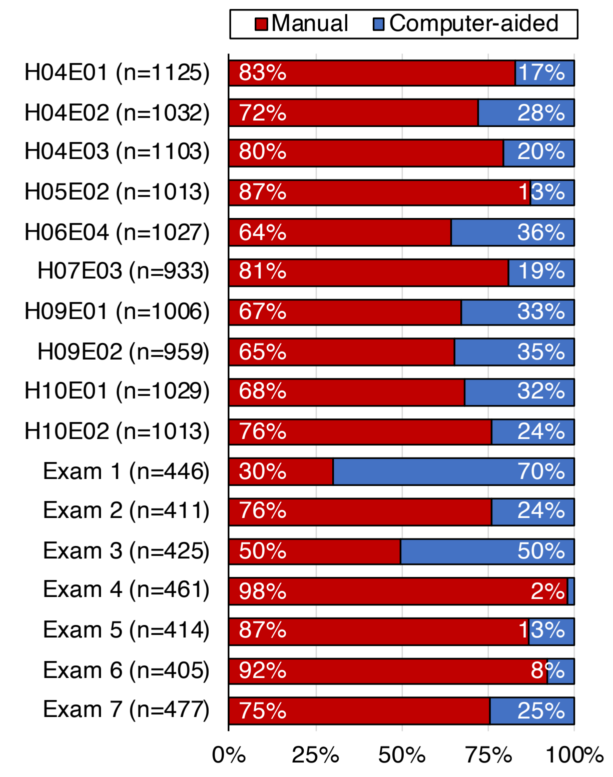


Figure 8. Exercises with their assessment ratios. Computer-aided reviews received automated grading suggestions which were reviewed by a teacher. On average, 26% reviews were computer-aided.

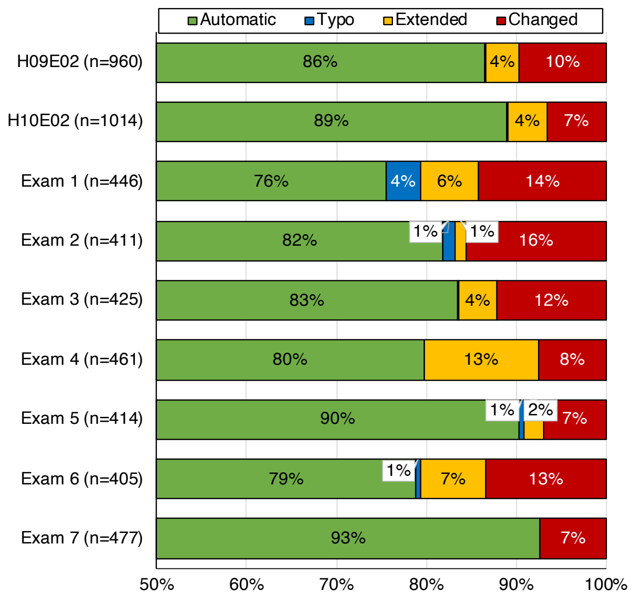


Figure 9. On average, 85% of computer-aided feedback comments remained unchanged (green) or only included minor typo fixes (blue). Furthermore, 5% were extended (yellow), and 10% were changed (red).

between three aspects of validity: internal validity, external validity, and construct validity [29].

Internal Validity: The accuracy of the feedback suggestions is measured by the acceptance of the teacher. A second review from a control teacher would allow for a more accurate measurement of accuracy. The teacher might be biased to confirm a feedback suggestion as it requires less effort than providing a new comment. We noticed that most teachers took the review of the automatic feedback suggestions seriously, but we cannot guarantee that some of the 49 involved teaching assistants failed to fully review the automatic feedback suggestions.

Two of the authors of this paper have been involved in teaching the course SE1 and might have influenced the empirical evaluation. However, we tried to clearly separate the research and instructor perspective. Two additional instructors have been involved in the course SE1 who are not authors of this paper, and the third author of the paper reviewed the results carefully without being involved in the course. In addition, we observed similar results in the second course, which was taught by an independent instructor who was not involved in the research.

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.

Construct Validity: The validity of the ratings might be affected by the wording of the question and by 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 had a good quality. Another limitation could be the fact 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 3 on the quality of the ratings as anecdotal evidence.

Discussion

The review coverage of Athene is higher for exercises that do not ask for examples, but rather require students to work based on a given example. In the exercises `Exam 1` and `Exam 3`, students were asked to extract requirements or use cases from a given problem statement. In those exercises, the coverage was above the average with 70% and 50%, respectively. 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 SE1 homework exercises, have lower review coverage of between 17% and 36%. This may be due to the increased variability of answers with different examples. As Athene tries to find similar text segments, it is more difficult to find a group with shared segments as students can describe all possible examples. Therefore, it is less likely to find reusable feedback among students.

Athene reuses reviews from teachers. The quality of the feedback suggestions depends on the quality of the manual feedback provided during the teacher reviews. If teachers provide incorrect manual feedback, Athene will not be able to pro-

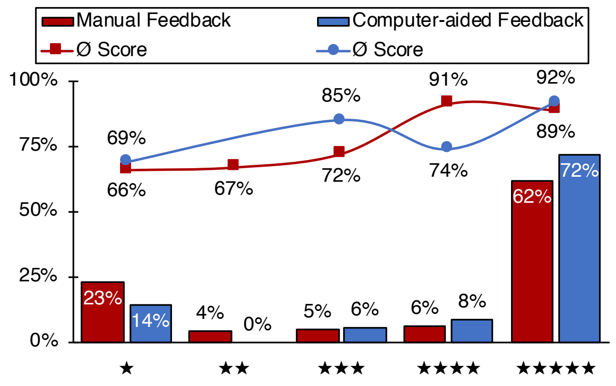


Figure 10. All ratings for SE1 homework (HXX) exercises by star rating. In this figure, ratings are grouped by the assessment type *Manual* ($n = 325$) or *Computer-aided* ($n = 71$). The average score in percent is depicted per rating and assessment type. In the study, 396 out of $10,240$ reviews were rated by students.

vide correct feedback suggestions. In the example of SE1, the teachers 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 the Related Work section suggest comparing answers only with a sample solution provided by an instructor [25], thus reducing the variability in the solution space, which might limit the creativity of the students. However, creativity is an important aspect in software engineering education [14].

CONCLUSION

This paper presents three main contributions:

1. The machine-learning based **approach** CoFee was presented to suggest feedback for textual exercises. The approach is based on segmentation and similarity-based clustering. It reuses feedback on segments within the same cluster and learns which aspects of student answers are correct during the assessment.
2. Athene, a **reference implementation** of CoFee, using the ELMo language model and the HDBSCAN clustering algorithm was presented. Athene is integrated into Artemis and published as open-source software under the MIT license.[7]
3. An **empirical evaluation** of Athene in two courses with $2,300$ students and 53 teachers in 17 textual exercises was conducted. The results of the quantitative evaluation in these exercises show that Athene can suggest up to 70% of the feedback with an average accuracy of 85%. Ratings provide first indications that the quality improves when compared to purely manual assessments.

The evaluation also shows that these numbers depend on the type of the textual exercise and on the variability of the possible solutions. A higher variance in correct solutions leads to less coverage because of fewer similarities in the student answers.

Athene does not require training data before the reviewing process 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. However, when reusing past exercises, Athene could profit from additional knowledge captured in these reviews. Future work needs to evaluate whether training data from the same exercise in previous years can improve the coverage or accuracy of feedback suggestions.

[7] Athene: https://github.com/ls1intum/Athene

REFERENCES

[1] Carlos Alario-Hoyos, Carlos Kloos, Iria Estévez-Ayres, Carmen Fernández-Panadero, Jorge Blasco, Sergio Pastrana, and J Villena-Román. 2016. Interactive activities: the key to learning programming with MOOCs. *European Stakeholder Summit on Experiences and Best Practices in and Around MOOCs* 319 (2016).

[2] Lorin W. Anderson, David R. Krathwohl, Peter W. Airasian, Kathleen A. Cruikshank, Richard E. Mayer, Paul R. Pintrich, James Raths, and Merlin C. Wittrock. 2001. *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*. Longmans Green.

[3] Sumit Basu, Chuck Jacobs, and Lucy Vanderwende. 2013. Powergrading: a clustering approach to amplify human effort for short answer grading. *Transactions of the Association for Computational Linguistics* 1 (2013), 391–402.

[4] Jan Philip Bernius. 2021. Toward Computer-Aided Assessment of Textual Exercises in Very Large Courses. In *52nd ACM Technical Symposium on Computer Science Education (SIGCSE '21)*. 1386.

[5] Jan Philip Bernius and Bernd Bruegge. 2019. Toward the Automatic Assessment of Text Exercises. In *2nd Workshop on Innovative Software Engineering Education*. Stuttgart, Germany, 19–22.

[6] Jan Philip Bernius, Anna Kovaleva, and Bernd Bruegge. 2020a. Segmenting Student Answers to Textual Exercises Based on Topic Modeling. In *17th Workshop Software Engineering im Unterricht der Hochschulen (SEUH)*. Stuttgart, Germany, 72–73.

[7] Jan Philip Bernius, Anna Kovaleva, Stephan Krusche, and Bernd Bruegge. 2020b. Towards the Automation of Grading Textual Student Submissions to Open-ended Questions. In *European Conference on Software Engineering Education*. ACM, 61–70.

[8] John Biggs. 2003. Aligning teaching and assessing to course objectives. *Teaching and learning in higher education: New trends and innovations* 2 (2003), 13–17.

[9] Benjamin S. Bloom, Max D. Engelhart, Edward J. Furst, Walker H. Hill, and David R. Krathwohl. 1956. *Taxonomy of educational objectives. The classification*

of educational goals. Handbook 1: Cognitive domain. Longmans Green.

[10] Richard P. Feynman. 1994. *Six Easy Pieces*.

[11] Richard Higgins, Peter Hartley, and Alan Skelton. 2002. The conscientious consumer: Reconsidering the role of assessment feedback in student learning. *Studies in higher education* 27, 1 (2002), 53–64.

[12] Alastair Irons. 2007. *Enhancing learning through formative assessment and feedback*. Routledge.

[13] Paul Kirschner, John Sweller, and Richard Clark. 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, 2 (2006), 75–86.

[14] Stephan Krusche, Bernd Bruegge, Irina Camilleri, Kirill Krinkin, Andreas Seitz, and Cecil Wöbker. 2017a. Chaordic Learning: A Case Study. In *39th International Conference on Software Engineering: Software Engineering Education and Training*. IEEE, 87–96.

[15] Stephan Krusche and Andreas Seitz. 2018. ArTEMiS: An Automatic Assessment Management System for Interactive Learning. In *49th ACM Technical Symposium on Computer Science Education (SIGCSE)*. 284–289.

[16] Stephan Krusche and Andreas Seitz. 2019. Increasing the Interactivity in Software Engineering MOOCs - A Case Study. In *52nd Hawaii International Conference on System Sciences*. 1–10.

[17] Stephan Krusche, Andreas Seitz, Jürgen Börstler, and Bernd Bruegge. 2017b. Interactive learning: Increasing student participation through shorter exercise cycles. In *19th Australasian Computing Education Conference*. ACM, 17–26.

[18] Stephan Krusche, Nadine von Frankenberg, and Sami Afifi. 2017c. Experiences of a Software Engineering Course based on Interactive Learning. In *Tagungsband des 15. Workshops Software Engineering im Unterricht der Hochschulen (SEUH)*. CEUR, 32–40.

[19] Stephan Krusche, Nadine von Frankenberg, Lara Marie Reimer, and Bernd Bruegge. 2020. An interactive learning method to engage students in modeling. In *International Conference on Software Engineering: Software Engineering Education and Training*. 12–22.

[20] Vladimir I. Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. *Soviet Physics-Doklady* 10, 8 (1966), 707–710.

[21] Yang Li and Tao Yang. 2018. *Word Embedding for Understanding Natural Language: A Survey*. Springer International Publishing, Cham, 83–104.

[22] Leland McInnes and John Healy. 2017. Accelerated Hierarchical Density Based Clustering. In *International Conference on Data Mining Workshops*. 33–42.

[23] Tom Mitchell, Terry Russell, Peter Broomhead, and Nicola Aldridge. 2002. Towards robust computerised marking of free-text responses. (2002).

[24] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Doha, Qatar, 1532–1543.

[25] Diana Perez, Alfio Gliozzo, Carlo Strapparava, Enrique Alfonseca, Pilar Rodríguez, and Bernardo Magnini. 2005. Automatic Assessment of Students' Free-Text Answers Underpinned by the Combination of a BLEU-Inspired Algorithm and Latent Semantic Analysis. 358–363.

[26] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2227–2237.

[27] Karl Raimund Popper. 1972. *Objective Knowledge*.

[28] Juan Ramos. 2003. Using TF-IDF to Determine Word Relevance in Document Queries. In *1st instructional conference on machine learning*, Vol. 242. Piscataway, NJ, 133–142.

[29] Per Runeson, Martin Höst, Austen Rainer, and Björn Regnell. 2012. *Case Study Research in Software Engineering*. John Wiley & Sons, Inc.

[30] Arjun Singh, Sergey Karayev, Kevin Gutowski, and Pieter Abbeel. 2017. Gradescope: A Fast, Flexible, and Fair System for Scalable Assessment of Handwritten Work. In *4th Conference on Learning @ Scale*. ACM, 81–88.

[31] Jana Sukkarieh, Stephen G Pulman, and Nicholas Raikes. 2003. Automarking: using computational linguistics to score short, free-text responses. (2003).

[32] Reed Williams and Thomas Haladyna. 1982. Logical Operations for Generating Intended Questions (LOGIQ): A typology for higher level test items. *A technology for test-item writing* (1982), 161–186.

[33] William E. Winkler. 1990. String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. In *Section on Survey Research*. 354–359.

8 Machine Learning Based Feedback on Textual Student Answers in Large Courses

Publication [BKB22] has been published as a **peer-reviewed journal article**:

Authors:	Jan Philip Bernius, Stephan Krusche, and Bernd Bruegge
Journal:	Computers and Education: Artificial Intelligence
ISSN:	2666-920X
Publisher:	Elsevier BV
Volume:	3
Year:	2022
DOI:	10.1016/j.caeai.2022.100081
Review:	Peer Reviewed (2 Reviewers)

Summary The paper formalizes the research methodology of the dissertation using the Wieringa design science process. The problem investigation of large courses resulted in the treatment design of the Computer-aided Feedback for textual exercises (CoFee) framework. The treatment validation of CoFee was conducted in a small laboratory experiment in 2019, which demonstrated that the grading overhead could be reduced by 85%. This treatment validation already confirmed the feasibility of automating the grading process for problem-solving exercises.

The treatment was implemented in the Athena reference implementation The top-level design of Athena consists of four components Segmentation, Language Embedding, Clustering, and Feedback Engine, which are described in detail in the paper.

The Athena implementation was then evaluated in two design cycles using a set of three courses with up to $2,200$ enrolled students per course. Data were collected from 34 exercises offered in each of the courses. Athena suggested feedback for 45% of the submissions. 92% (Positive Predictive Value) of these suggestions were precise and, therefore, accepted by the instructors.

Contributions **J.P. B.** conceptualized and formalized the architecture and dynamic behavior of CoFee. **J.P. B.** and S. K. formulated the problem investigation. **J.P. B.** derived the knowledge goals according to Wieringa's design science method and conducted the literature review. **J.P. B.** developed Athenas system design and implemented Athena together with the people mentioned in the articles Acknowledgments section. **J.P. B.** developed the implementation evaluations study design, collected the data, performed the statistical analysis, and described the findings and results. **J.P. B.** wrote and visualized the paper. B. B. provided feedback and helped improving the manuscript. S. K. and B. B. reviewed the paper.

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

A R T I C L E I N F O

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

A B S T R A C T

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

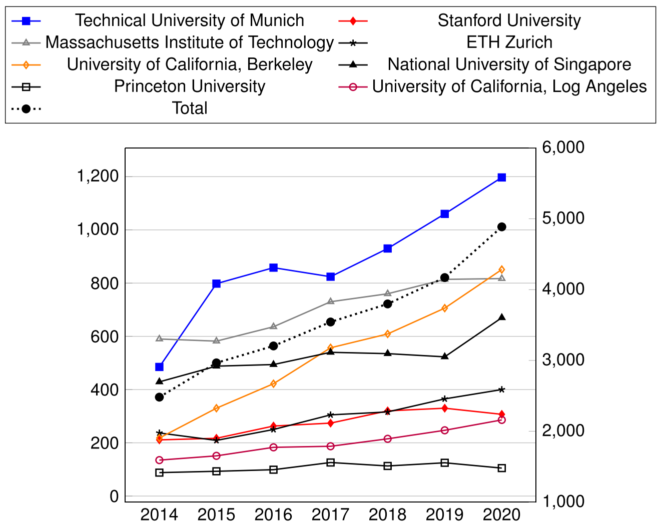


Fig. 1. The number of computer science degrees (bachelor's and master's) conferred per year by renowned universities in the area has steadily increased over the last decade. Data was collected from statistics published by the universities. The left y-axis represents the number of degrees per university.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

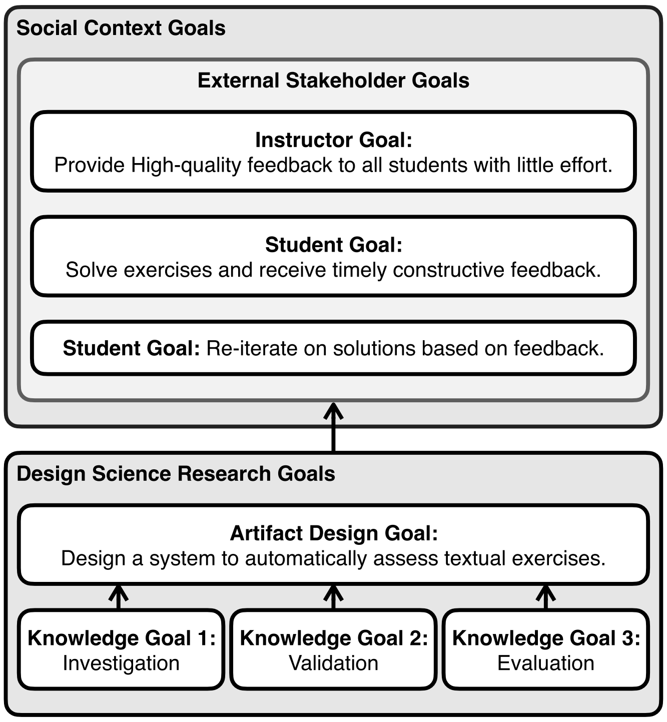

Social Context Goals
External Stakeholder Goals
Instructor Goal:
Provide High-quality feedback to all students with little effort.
Student Goal:
Solve exercises and receive timely constructive feedback.
Student Goal: Re-iterate on solutions based on feedback.
Design Science Research Goals
Artifact Design Goal:
Design a system to automatically assess textual exercises.
Knowledge Goal 1: Investigation
Knowledge Goal 2: Validation
Knowledge Goal 3: Evaluation

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:

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

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

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 <u>co</u>mputer-aided <u>fee</u>dback 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 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.

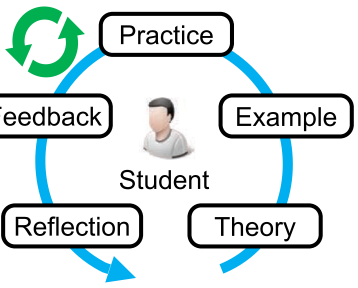


Fig. 3. Interactive learning puts the individual student into the core of the learning activity and follows an iterative process that is conducted multiple times in lectures.

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

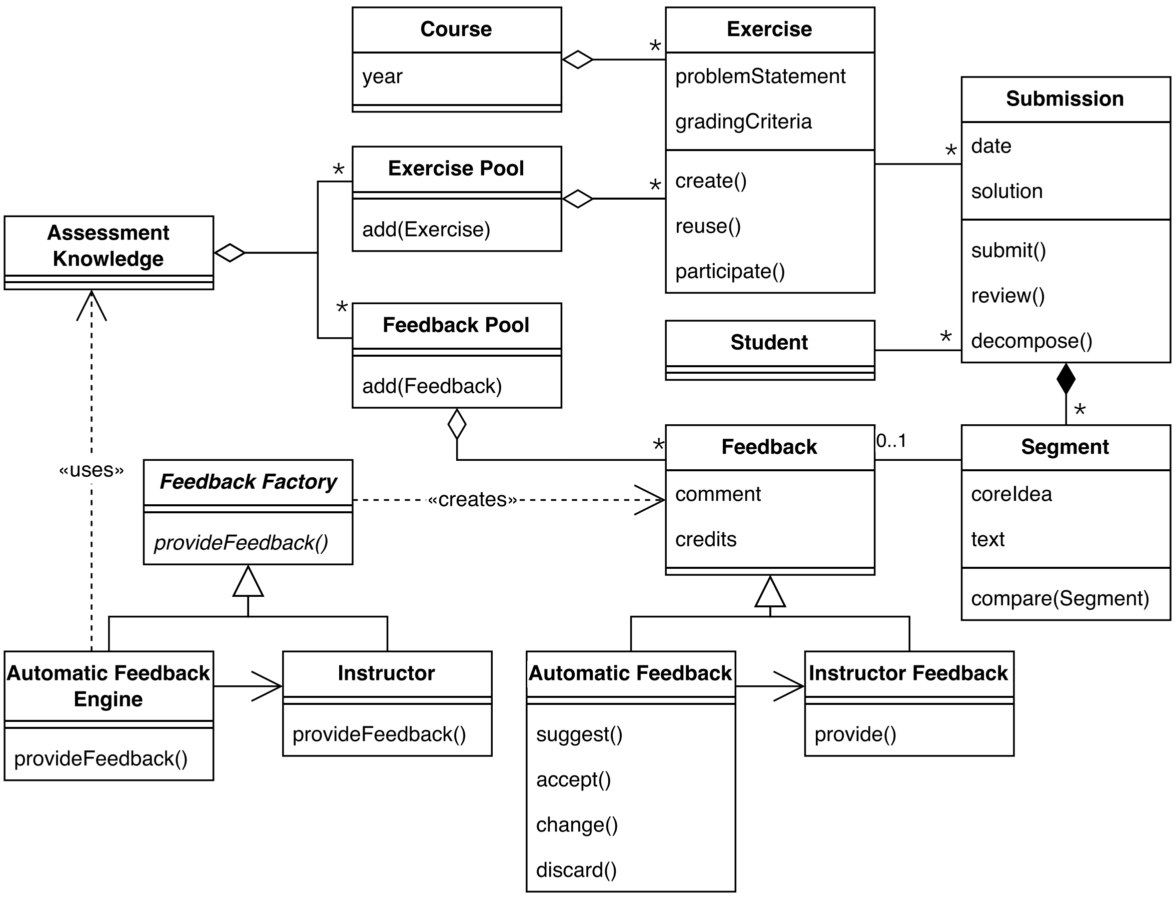


Fig. 4. Analysis Object Model of the CoFee framework. The model describes the system from the stakeholder's point of view and illustrates the concepts visible to the stakeholder (Bruegge & Dutoit, 2009) (UML Class Diagram).

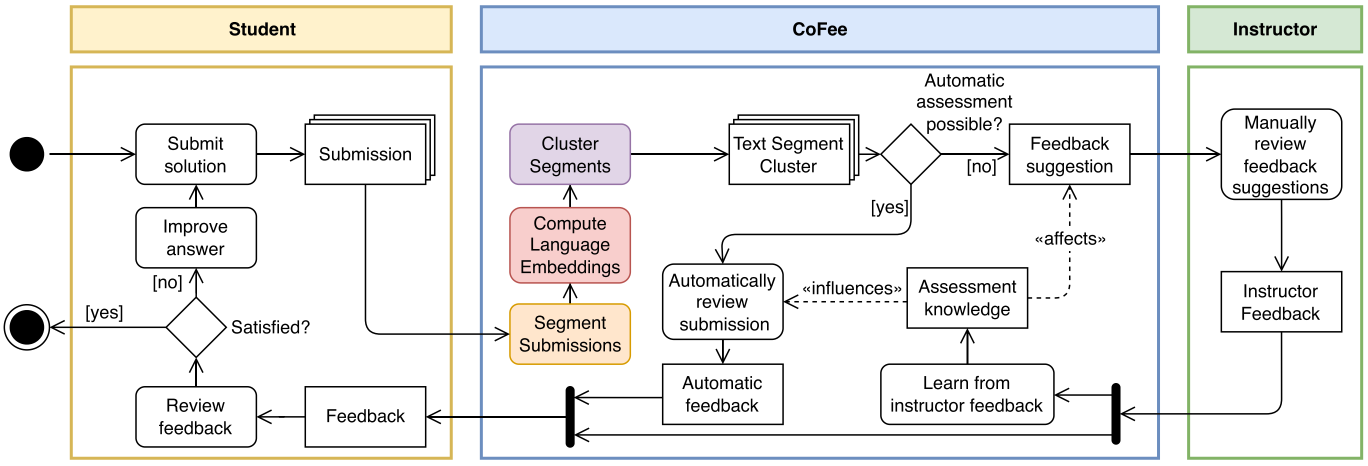


Fig. 5. Workflow of automatic assessment of submissions to textual exercises based on the manual feedback of instructors. CoFee analyzes manual assessments and generates knowledge for the suggestion of computer-aided (automatic) feedback (UML activity diagram).

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.

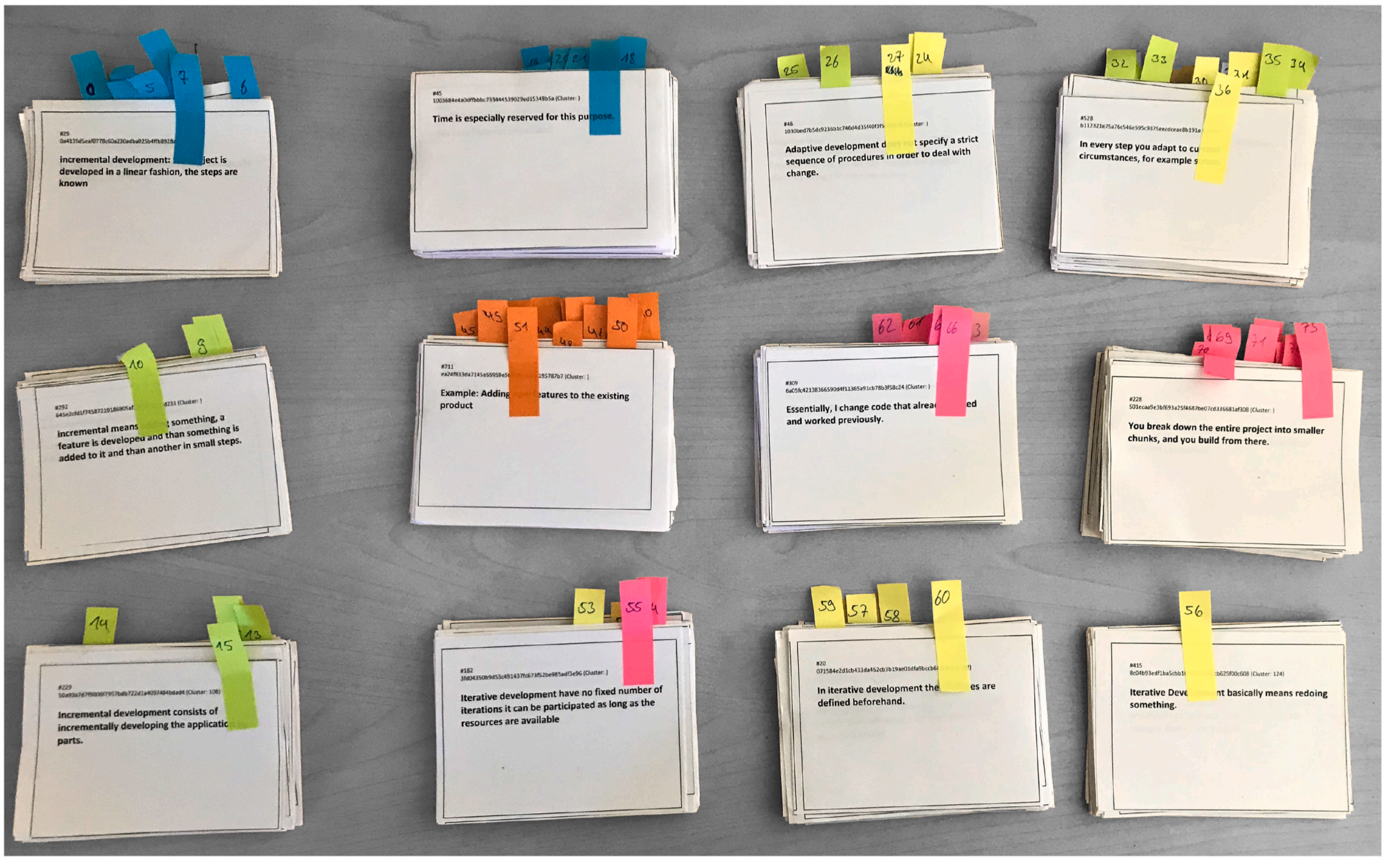


Fig. 6. 760 text segments clustered by hand into 75 clusters.

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.

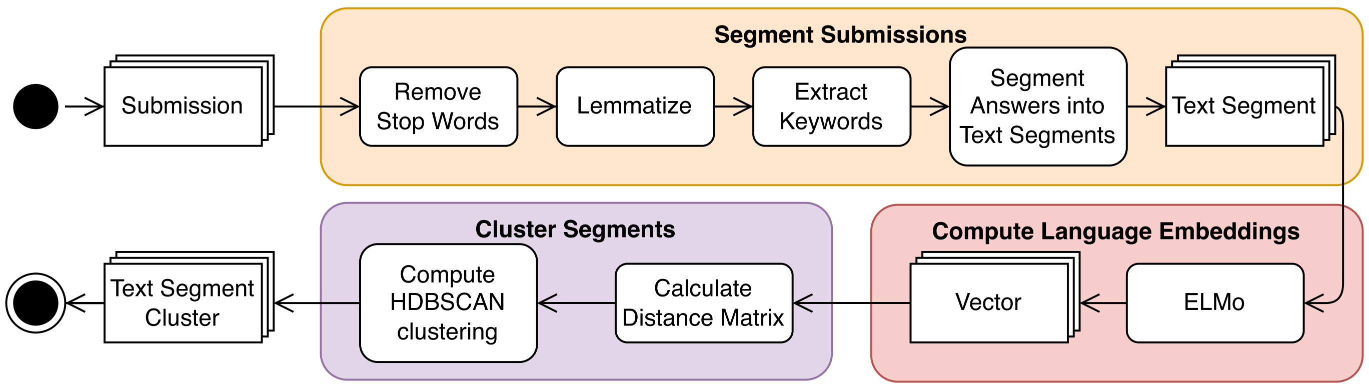


Fig. 7. Overview of the machine learning activities making up the "Segment Submissions", "Compute Language Embeddings", and "Cluster Segments" activities in Fig. 5. These are used to extract text segments and build text clusters for scoring and similarity analysis (UML activity diagram).

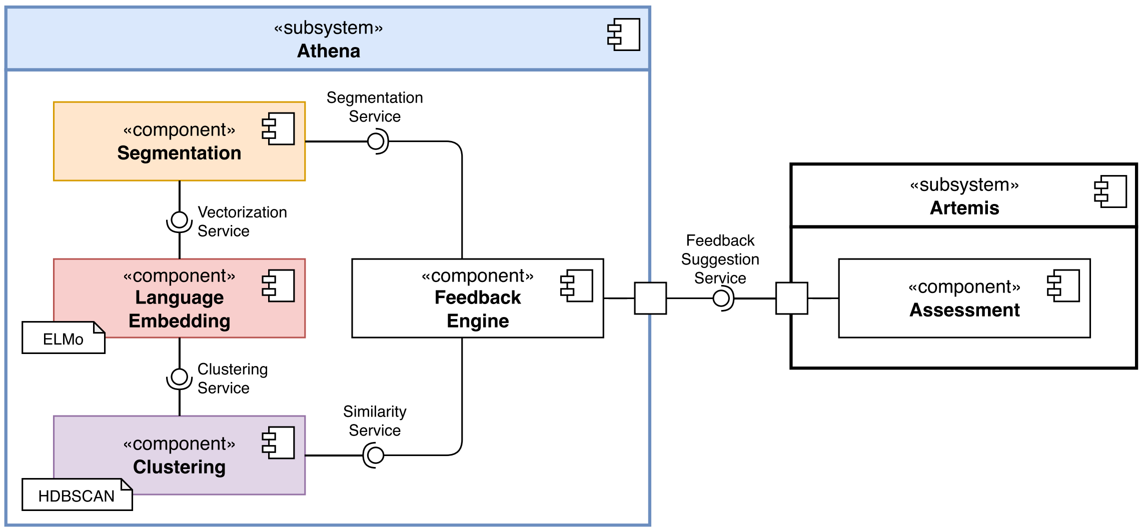
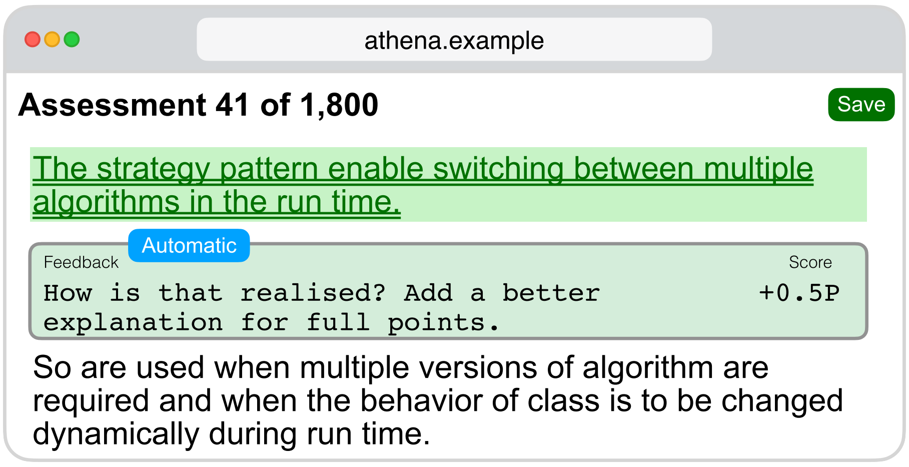


Fig. 8. Top-level design of the Athena system. Athena is composed of four components: Segmentation, Language Embedding, and Clustering implement the machine learning activities depicted in Fig. 7. The Feedback Engine acts as the facade to Artemis and offers an API that the Assessment component uses to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


athena.example

Assessment 41 of 1,800 Save

The strategy pattern enable switching between multiple algorithms in the run time.

Automatic
Feedback Score
How is that realised? Add a better explanation for full points. +0.5P

So are used when multiple versions of algorithm are required and when the behavior of class is to be changed dynamically during run time.

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*

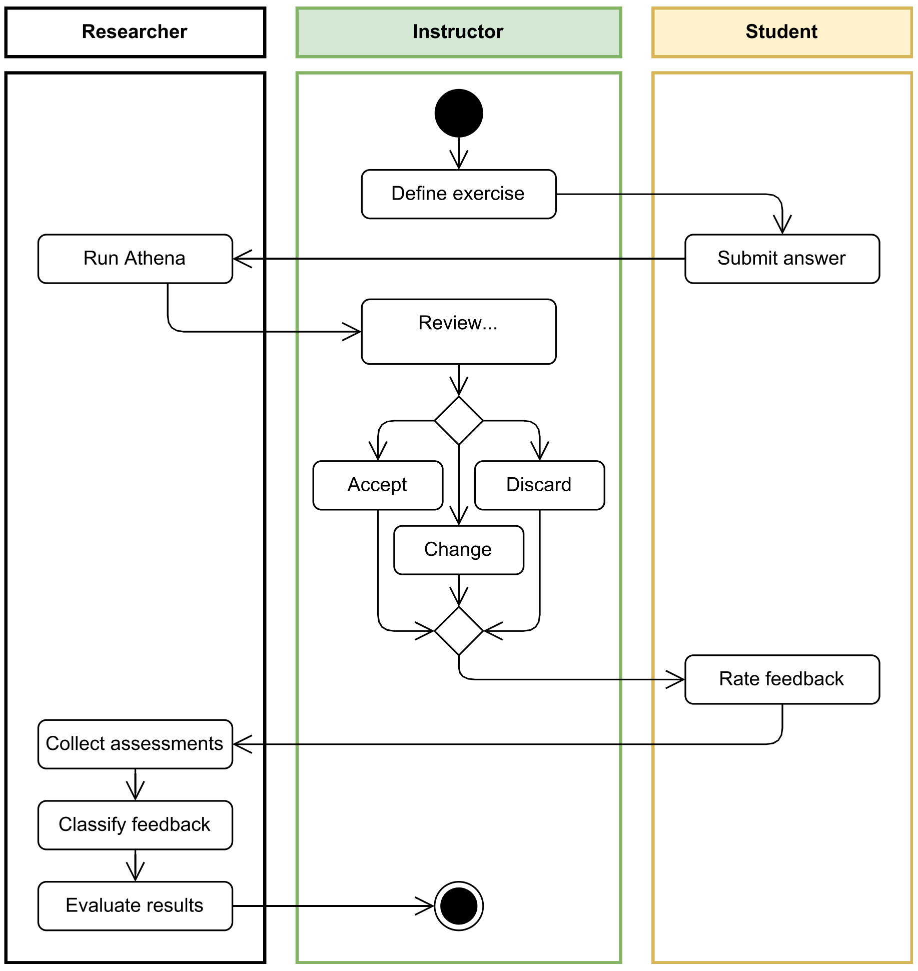


Fig. 10. Research approach depicted with the involved actors and flow of events (UML activity diagram).

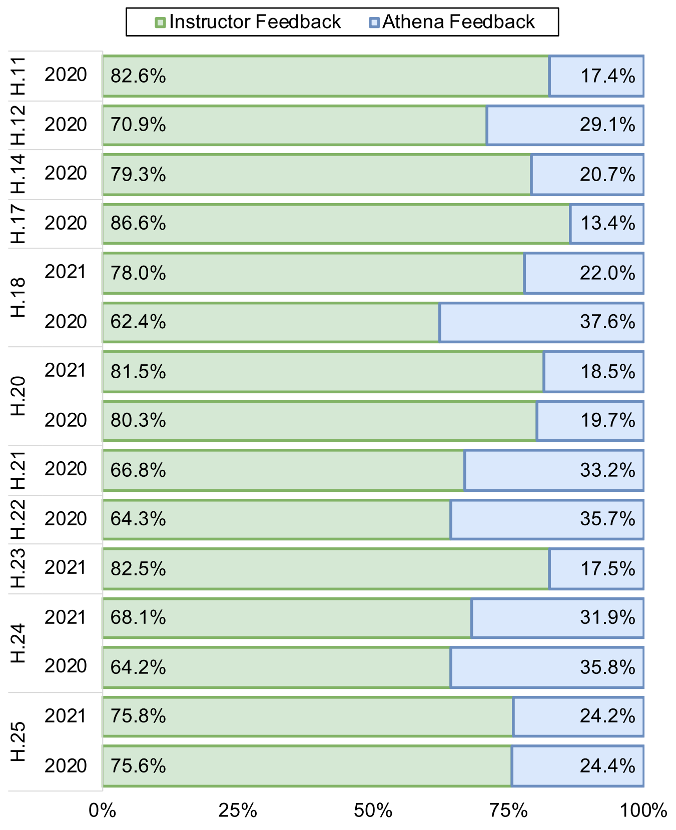


Fig. 11. Homework exercises with their assessment ratios. Athena feedback reviews received automated suggestions which were reviewed by instructors. On average, 25% of all homework assessments were computer-aided.

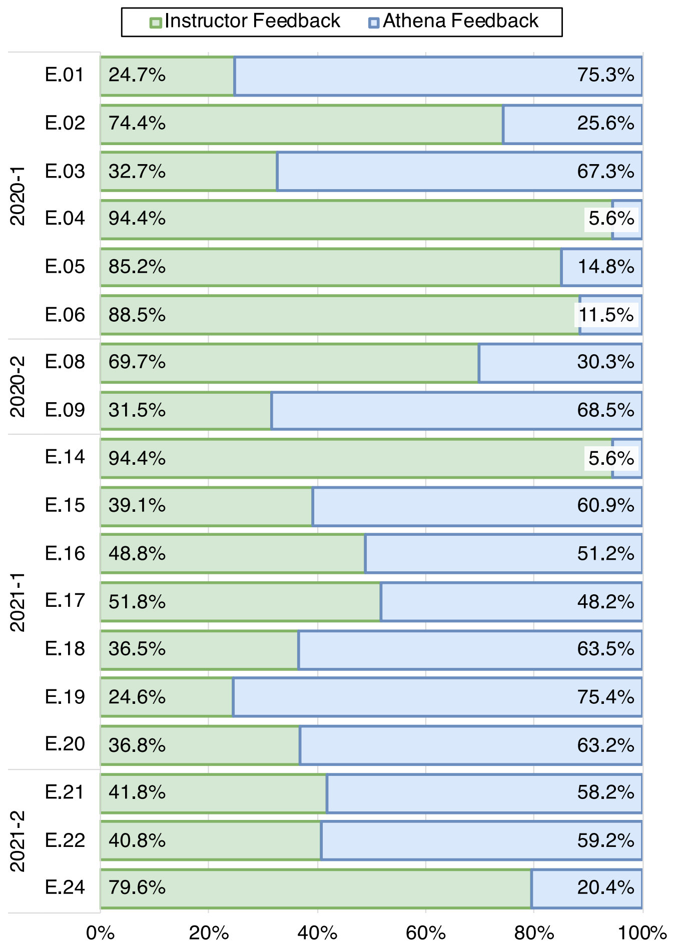


Fig. 12. Exam exercises with their assessment ratios. Athena feedback reviews received automated suggestions which were reviewed by instructors. On average, 54% of all exam assessments were computer-aided.

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).

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

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*** $\left(\overline{PPV} = 92\%\right)$.

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%

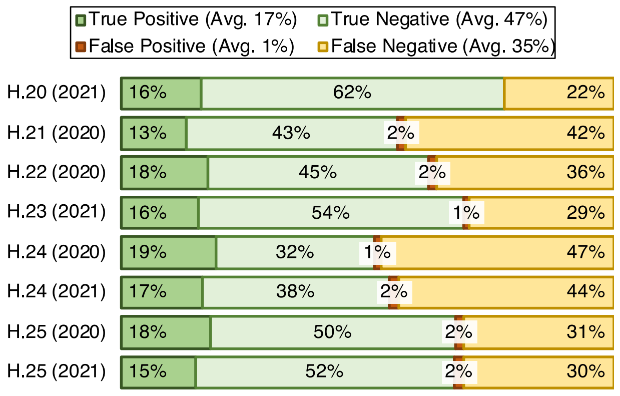


Fig. 13. Visualization of Binary Classification (Table 2) in percent.

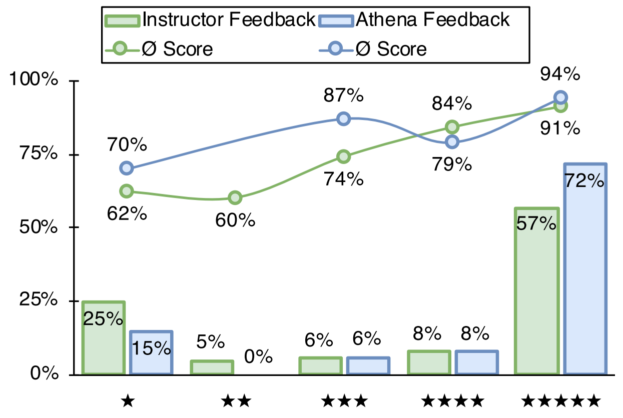


Fig. 14. All ratings for ISE homework exercises by star rating. In this figure, ratings are grouped by *Instructor Feedback* ($n = 428$) or *Athena Feedback* ($n = 102$). The average score in percent is depicted per rating and assessment type. In the study, 530 out of 15, 868 reviews were rated by students.

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.

9 Conclusion & Outlook

This dissertation presented an approach to automate the assessment of textual exercises, thereby reducing the assessment efforts for instructors of large classes. The **four main scientific contributions** of this dissertation are:

A formalization and the design of the Computer-aided Feedback for textual exercises (CoFee) framework [BB19; BKB21; BKB22] outlines how to capture assessment knowledge and automatically suggests feedback based on machine learning. The framework employs segment-based grading and reuses feedback based on segment similarity. CoFee can identify similar segments which should receive similar feedback based on a three-fold process consisting of segmentation of answers, intermediate representations based on language embeddings, and clustering. The treatment validation confirmed the frameworks' validity in a laboratory experiment and found that CoFee can reduce the instructors grading effort by 85%.

The algorithm for automatic segmentation based on topic modeling [Ber+20] identifies relevant topics discussed within the scope of an exercise and represents topics using keywords. The segmentation algorithm is based on a divide-and-conquer approach to decompose student answers into topically coherent segments. Phrases and sentences are merged until a topic shift occurs. The evaluation confirms that segments produced by the algorithm are better suited for assessment than sentences split by interpunctuation.

The reference implementation of the CoFee design is called "Athena" [BKB21; BKB22] and demonstrates how to design and build a system that automatically assesses textual exercises. Athena uses the segmentation algorithm [Ber+20] to decompose answers, the ELMo language model to capture core ideas of segments, and Hierarchical Density-Based Spatial Clustering of Applications with Noise (HDBSCAN) clustering to identify groups of similar segments. Athena is open-source software published under the MIT license and integrated into the Artemis learning management system.

The empirical evaluations [BKB21; BKB22] describe the usage of Athena in several large-scale software engineering courses at the Technical University of Munich (TUM) between 2019 and 2021. Up to 2,200 students and up to 68 instructors participated in the courses. 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's 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 contributions are published in four publications in an international workshop [BB19], two international conferences [Ber+20; BKB21], and an international journal [BKB22].

CoFee models a closed feedback loop with continuous feedback for students solving an exercise. Athena does not implement this closed feedback loop, which would allow students to submit their solutions multiple times, including intermediate solutions. Allowing multiple submissions for every exercise will significantly increase the grading workload. In the future, Athena can help reduce the grading workload down to new knowledge submitted in a new iteration and automate the grading of common or repeated mistakes.

The nature of the hybrid grading process depends on a human grader to review and complete the feedback suggestions from Athena. In the future, Athena should enable real-time evaluations of submitted answers by removing humans from the grading loop. Furthermore, humans should only be involved in grading if the trained assessment knowledge is insufficient to provide confident feedback.

Fast-changing paradigms in computer science make it hard to keep materials and exercises up to date. Therefore, Athena should support evolving curriculum design by offering automated grading based on partial knowledge and incremental training of the assessment knowledge. Examples of evolving exercises are new problem statements to apply concepts or new interpretations of concepts and trade-offs.

Another challenge is to evaluate CoFee and Athena in a different domain. The Athena system was built independently from specific domains and should support exercises from all domains. However, evaluations of Athena focused on exercises related to computer science. Therefore, future evaluations on the precision and accuracy of Athena in other domains are needed.

List of Figures

List of Tables

Acronyms

CoFee	<u>Co</u>mputer-aided <u>Fee</u>dback for textual exercises.
DBSCAN	Density-Based Spatial Clustering of Applications with Noise.
EIST	Introduction to Software Engineering.
ELMo	Embeddings from Language Models.
HDBSCAN	Hierarchical Density-Based Spatial Clustering of Applications with Noise.
ISE	Introduction to Software Engineering.
LDA	Latent Dirichlet Allocation.
LMS	Learning Management System.
LSA	Latent Semantic Analysis.
LSTM	Long Short-Term Memory.
MOOC	Massive Open Online Course.
NLP	Natural Language Processing.
POM	Project Organisation and Management in Software Engineering.
PPV	Positive Predictive Value.
PSE	Patterns in Software Engineering.
TF-IDF	Term Frequency-Inverse Document Frequency.
TNR	True Negative Rate.
TPR	True Positive Rate.
TUM	Technical University of Munich.
WMT	Workshop on Machine Translation.

Bibliography

[Ala18] Jay Alammar. *The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning).* Licensed under CC BY-NC-SA 2.0. December 2018. URL: https://jalammar.github.io/illustrated-bert/ (visited on 20.5.2022).

[AK97] Vicki H. Allan and Mary Veronica Kolesar. "Teaching Computer Science: A Problem Solving Approach That Works." In: *SIGCUE Outlook* 25.1–2 (January 1997), pp. 2–10. ISSN: 0163-5735. DOI: 10.1145/274375.274376.

[And+01] Lorin W. Anderson, David R. Krathwohl, Peter W. Airasian, Kathleen A. Cruikshank, Richard E. Mayer, Paul R. Pintrich, James Raths, and Merlin C. Wittrock. *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives.* Longmans Green, 2001. ISBN: 9780801319037.

[BBL97] Doug Beeferman, Adam Berger, and John Lafferty. "Text Segmentation Using Exponential Models." In: *2nd Conference on Empirical Methods in Natural Language Processing.* 1997. URL: https://aclanthology.org/W97-0304.

[Ber21] Jan Philip Bernius. "Toward Computer-Aided Assessment of Textual Exercises in Very Large Courses." In: *52nd ACM Technical Symposium on Computer Science Education.* SIGCSE '21. Toronto, ON, Canada: Association for Computing Machinery (ACM), March 2021, p. 1386. DOI: 10.1145/3408877.3439703.

[BB19] Jan Philip Bernius and Bernd Bruegge. "Toward the Automatic Assessment of Text Exercises." In: *2nd Workshop on Innovative Software Engineering Education.* ISEE '19. Stuttgart, Germany: CEUR-WS.org, February 2019, pp. 19–22. URL: http://ceur-ws.org/Vol-2308/isee2019paper04.pdf.

[BKB20] Jan Philip Bernius, Anna Kovaleva, and Bernd Bruegge. "Segmenting Student Answers to Textual Exercises Based on Topic Modeling." In: *17th Workshop on Software Engineering im Unterricht der Hochschulen.* SEUH '20. Innsbruck, Austria: CEUR-WS.org, February 2020, pp. 72–73. URL: http://ceur-ws.org/Vol-2531/poster03.pdf.

[Ber+20] Jan Philip Bernius, Anna Kovaleva, Stephan Krusche, and Bernd Bruegge. "Towards the Automation of Grading Textual Student Submissions to Open-ended Questions." In: *4th European Conference of Software Engineering Education*. ECSEE '20. Seeon, Germany: Association for Computing Machinery (ACM), June 2020, pp. 61–70. ISBN: 9781450377522. DOI: 10.1145/3396802.3396805.

[BKB21] Jan Philip Bernius, Stephan Krusche, and Bernd Bruegge. "A Machine Learning Approach for Suggesting Feedback in Textual Exercises in Large Courses." In: *8th ACM Conference on Learning @ Scale*. L@S '21. Potsdam, Germany: Association for Computing Machinery (ACM), June 2021, pp. 173–182. ISBN: 9781450382151. DOI: 10.1145/3430895.3460135.

[BKB22] Jan Philip Bernius, Stephan Krusche, and Bernd Bruegge. "Machine Learning Based Feedback on Textual Student Answers in Large Courses." In: *Computers and Education: Artificial Intelligence* 3 (June 2022). ISSN: 2666-920X. DOI: 10.1016/j.caeai.2022.100081.

[Big03] John Biggs. "Aligning teaching and assessing to course objectives." In: *Teaching and learning in higher education: New trends and innovations* 2 (2003), pp. 13–17.

[BNJ03] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. "Latent Dirichlet Allocation." In: *Journal of Machine Learning Research* 3 (2003), pp. 993–1022. URL: http://jmlr.org/papers/v3/blei03a.html.

[BE91] Charles C Bonwell and James A Eison. *Active Learning: Creating Excitement in the Classrom*. ASHE-ERIC Higher Education Reports. Washington, DC, USA: School of Education & Human Development, The George Washington University, January 1991. ISBN: 1-878380-08-7.

[BCT02] Thorsten Brants, Francine Chen, and Ioannis Tsochantaridis. "Topic-based document segmentation with probabilistic latent semantic analysis." In: *11th international conference on Information and knowledge management*. CIKM '02. Association for Computing Machinery (ACM), 2002. DOI: 10.1145/584792.584829.

[BD09] Bernd Bruegge and Allen H Dutoit. *Object Oriented Software Engineering Using UML, Patterns, and Java*. Prentice Hall, 2009.

[BGS15] Steven Burrows, Iryna Gurevych, and Benno Stein. "The eras and trends of automatic short answer grading." In: *International Journal of Artificial Intelligence in Education* 25.1 (2015), pp. 60–117.

[CW14] Erik Cambria and Bebo White. "Jumping NLP Curves: A Review of Natural Language Processing Research [Review Article]." In: *IEEE Computational Intelligence Magazine* 9.2 (2014), pp. 48–57. DOI: `10.1109/MCI.2014.2307227`.

[Cam+19] Ricardo J. G. B. Campello, Peer Kröger, Jörg Sander, and Arthur Zimek. "Density-based clustering." In: *WIREs Data Mining and Knowledge Discovery* 10.2 (October 2019). DOI: `10.1002/widm.1343`.

[Cam+15] Ricardo J. G. B. Campello, Davoud Moulavi, Arthur Zimek, and Jörg Sander. "Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection." In: *ACM Transactions on Knowledge Discovery from Data* 10.1 (July 2015), pp. 1–51. DOI: `10.1145/2733381`.

[Car80] Jaime G. Carbonell. "DELTA-MIN: A Search-Control Method for Information-Gathering Problems." In: *1st Annual National Conference on Artificial Intelligence.* Stanford University, CA, USA: AAAI Press/MIT Press, August 1980, pp. 124–127. URL: `http://www.aaai.org/Library/AAAI/1980/aaai80-036.php`.

[CCG81] Jaime G. Carbonell, Richard E. Cullingford, and Anatole V. Gershman. "Steps Toward Knowledge-Based Machine Translation." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-3.4 (1981), pp. 376–392. DOI: `10.1109/TPAMI.1981.4767124`.

[Car+97] Jaime G. Carbonell, Yiming Yang, Robert E. Frederking, Ralf D. Brown, Yibing Geng, and Danny Lee. "Translingual Information Retrieval: A Comparative Evaluation." In: *15th International Joint Conference on Artificial Intelligence.* IJCAI 97. Nagoya, Japan: Morgan Kaufmann, August 1997, pp. 708–715.

[Car+10] David Carless, Diane Salter, Min Yang, and Joy Lam. "Developing sustainable feedback practices." In: *Studies in Higher Education* 36.4 (November 2010), pp. 395–407. DOI: `10.1080/03075071003642449`.

[Cat43] Raymond B. Cattell. "The description of personality: basic traits resolved into clusters." In: *Journal of Abnormal and Social Psychology* 38.4 (October 1943), pp. 476–506. DOI: `10.1037/h0054116`.

[CYY16] Qiuxing Chen, Lixiu Yao, and Jie Yang. "Short text classification based on LDA topic model." In: *2016 International Conference on Audio, Language and Image Processing.* ICALIP. IEEE, July 2016. DOI: `10.1109/icalip.2016.7846525`.

[Cho00] Freddy Y. Y. Choi. "Advances in domain independent linear text segmentation." In: *1st Meeting of the North American Chapter of the Association for Computational Linguistics*. 2000. URL: `https://aclanthology.org/A00-2004`.

[CWM01] Freddy Y. Y. Choi, Peter Wiemer-Hastings, and Johanna Moore. "Latent Semantic Analysis for Text Segmentation." In: *2001 Conference on Empirical Methods in Natural Language Processing*. 2001. URL: `https://aclanthology.org/W01-0514`.

[Cho56] Noam Chomsky. "Three models for the description of language." In: *IRE Transactions on Information Theory* 2.3 (1956), pp. 113–124. DOI: `10.1109/TIT.1956.1056813`.

[DKM98] Fadi P. Deek, Howard Kimmel, and James A. McHugh. "Pedagogical Changes in the Delivery of the First-Course in Computer Science: Problem Solving, Then Programming." In: *Journal of Engineering Education* 87.3 (July 1998), pp. 313–320. DOI: `10.1002/j.2168-9830.1998.tb00359.x`.

[Dev+19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: *019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (NAACL). Vol. 1 (Long Papers). Minneapolis, Minnesota: Association for Computational Linguistics (ACL), June 2019, pp. 4171–4186. DOI: `10.18653/v1/N19-1423`.

[DK32] H. E. Driver and A. L. Kroeber. "Quantitative Expression of Cultural Relationships." In: *American Archaeology and Ethnology* 31.4 (1932), pp. 211–256.

[Eis09] Jacob Eisenstein. "Hierarchical Text Segmentation from Multi-Scale Lexical Cohesion." In: *Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Boulder, Colorado: Association for Computational Linguistics (ACL), June 2009, pp. 353–361. URL: `https://aclanthology.org/N09-1040`.

[EB08] Jacob Eisenstein and Regina Barzilay. "Bayesian Unsupervised Topic Segmentation." In: *2008 Conference on Empirical Methods in Natural Language Processing*. Honolulu, Hawaii: Association for Computational Linguistics (ACL), October 2008, pp. 334–343. URL: `https://aclanthology.org/D08-1035`.

[EL90] David G. Elliman and Ian T Lancaster. "A review of segmentation and contextual analysis techniques for text recognition." In: *Pattern Recognition* 23.3-4 (January 1990), pp. 337–346. DOI: 10.1016/0031-3203(90)90021-c.

[Est+96] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise." In: *2nd International Conference on Knowledge Discovery and Data Mining.* KDD'96. Portland, Oregon: AAAI Press, 1996, pp. 226–231. URL: http://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf.

[Est02] Vladimir Estivill-Castro. "Why so many clustering algorithms." In: *ACM SIGKDD Explorations Newsletter* 4.1 (June 2002), pp. 65–75. DOI: 10.1145/568574.568575.

[Fey94] Richard P. Feynman. *Six Easy Pieces.* Basic Books, July 1994. ISBN: 978-0-2014-0825-6.

[FWA07] Dominik Flejter, Karol Wieloch, and Witold Abramowicz. "Unsupervised Methods of Topical Text Segmentation for Polish." In: *Annual Meeting of the Association for Computational Linguistics 2007 - Workshop on Balto-Slavonic Natural Language Processing: Information Extraction and Enabling Technologies.* ACL '07. Prague, Czech Republic: Association for Computational Linguistics (ACL), 2007, pp. 51–58.

[FP10] Scott Freeman and John W. Parks. "How Accurate Is Peer Grading?" In: *CBE—Life Sciences Education* 9.4 (December 2010), pp. 482–488. DOI: 10.1187/cbe.10-03-0017.

[Gao+10] Yang Gao, Li Zhou, Yong Zhang, Chunxiao Xing, Yigang Sun, and Xianzhong Zhu. "Sentiment classification for stock news." In: *5th International Conference on Pervasive Computing and Applications.* IEEE, December 2010. DOI: 10.1109/icpca.2010.5704082.

[GS05] Graham Gibbs and Claire Simpson. "Conditions Under Which Assessment Supports Students' Learning." In: *Learning and Teaching in Higher Education* 1 (2005), pp. 3–31. URL: http://eprints.glos.ac.uk/id/eprint/3609.

[GC98] Jade Goldstein and Jaime Carbonell. "Summarization: (1) Using MMR for Diversity- Based Reranking and (2) Evaluating Summaries." In: *TIPSTER Text Program Phase III Workshop.* Baltimore, Maryland, USA: Association for Computational Linguistics (ACL), October 1998, pp. 181–195. DOI: 10.3115/1119089.1119120. URL: https://aclanthology.org/X98-1025.

[Gol+99] Jade Goldstein, Mark Kantrowitz, Vibhu Mittal, and Jaime Carbonell. “Summarizing Text Documents: Sentence Selection and Evaluation Metrics.” In: *22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR ’99. Berkeley, California, USA: Association for Computing Machinery (ACM), 1999, pp. 121–128. ISBN: 1581130961. DOI: 10.1145/312624.312665.

[GL08] Victor Gonzalez-Barbone and Martin Llamas-Nistal. “eAssessment of open questions: An educator’s perspective.” In: *2008 38th Annual Frontiers in Education Conference.* IEEE. 2008, F2B–1.

[HH76] Michael Alexander Kirkwood Halliday and Ruqaiya Hasan. *Cohesion in English.* London, England: Longman, May 1976. ISBN: 9780582550414.

[Hea94] Marti A. Hearst. “Multi-Paragraph Segmentation Expository Text.” In: *32nd Annual Meeting of the Association for Computational Linguistics.* Las Cruces, New Mexico, USA: Association for Computational Linguistics (ACL), June 1994, pp. 9–16. DOI: 10.3115/981732.981734.

[Hea97] Marti A. Hearst. “Text Tiling: Segmenting Text into Multi-paragraph Subtopic Passages.” In: *Computational Linguistics* 23.1 (1997), pp. 33–64. URL: https://aclanthology.org/J97-1003.

[Hie09] Djoerd Hiemstra. “Language Models.” In: *Encyclopedia of Database Systems.* Springer US, 2009, pp. 1591–1594. DOI: 10.1007/978-0-387-39940-9_923.

[Hoa+18] Vu Cong Duy Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. “Iterative Back-Translation for Neural Machine Translation.” In: *2nd Workshop on Neural Machine Translation and Generation.* Melbourne, Australia: Association for Computational Linguistics (ACL), July 2018, pp. 18–24. DOI: 10.18653/v1/W18-2703.

[Hua+03] Xiangji Huang, Fuchun Peng, Dale Schuurmans, Nick Cercone, and Stephen E. Robertson. In: *Information Retrieval* 6.3/4 (2003), pp. 333–362. DOI: 10.1023/a:1026028229881.

[JB92] Anil K. Jain and Sushil Bhattacharjee. “Text segmentation using gabor filters for automatic document processing.” In: *Machine Vision and Applications* 5.3 (June 1992), pp. 169–184. DOI: 10.1007/bf02626996.

[JR10] Bernard J. Jansen and Soo Young Rieh. "The seventeen theoretical constructs of information searching and information retrieval." In: *Journal of the American Society for Information Science and Technology* 61.8 (2010), pp. 1517–1534. DOI: 10.1002/asi.21358.

[JHG13] Tansy Jessop, Yassein El Hakim, and Graham Gibbs. "The whole is greater than the sum of its parts: a large-scale study of students' learning in response to different programme assessment patterns." In: *Assessment & Evaluation in Higher Education* 39.1 (April 2013), pp. 73–88. DOI: 10.1080/02602938.2013.792108.

[JM09] Sally Jordan and Tom Mitchell. "e-Assessment for learning? The potential of short-answer free-text questions with tailored feedback." In: *British Journal of Educational Technology* 40.2 (2009), pp. 371–385.

[JCN13] S. Joty, G. Carenini, and R. T. Ng. "Topic Segmentation and Labeling in Asynchronous Conversations." In: *Journal of Artificial Intelligence Research* 47 (July 2013), pp. 521–573. DOI: 10.1613/jair.3940.

[KG17] Jan Knobloch and Enrico Gigantiello. "AMATI: Another Massive Audience Teaching Instrument." In: *15. Workshop on Software Engineering im Unterricht der Hochschulen.* SEUH '17. Hannover, Germany: CEUR-WS.org, February 2017, pp. 63–68. URL: http://ceur-ws.org/Vol-1790/paper07.pdf.

[KA10] DAVID R. Krahtwohl and LORIN W. Anderson. "Merlin C. Wittrock and the Revision of Bloom's Taxonomy." In: *Educational Psychologist* 45.1 (2010), pp. 64–65. DOI: 10.1080/00461520903433562.

[Kri+11] Hans-Peter Kriegel, Peer Kröger, Jörg Sander, and Arthur Zimek. "Density-based clustering." In: *WIREs Data Mining and Knowledge Discovery* 1.3 (April 2011), pp. 231–240. DOI: 10.1002/widm.30.

[KFA17] Stephan Krusche, Nadine von Frankenberg, and Sami Afifi. "Experiences of a Software Engineering Course based on Interactive Learning." In: *15. Workshop Software Engineering im Unterricht der Hochschulen.* SEUH '17. CEUR, 2017, pp. 32–40.

[Kru+20] Stephan Krusche, Nadine von Frankenberg, Lara Marie Reimer, and Bernd Bruegge. "An interactive learning method to engage students in modeling." In: *International Conference on Software Engineering: Software Engineering Education and Training.* 2020, pp. 12–22.

[KS19] Stephan Krusche and Andreas Seitz. “Increasing the Interactivity in Software Engineering MOOCs - A Case Study.” In: *52nd Hawaii International Conference on System Sciences, HICSS 2019, Grand Wailea, Maui, Hawaii, USA, January 8-11, 2019.* ScholarSpace, 2019, pp. 1–10. URL: `https://hdl.handle.net/10125/60197`.

[Kru+17] Stephan Krusche, Andreas Seitz, Jürgen Börstler, and Bernd Bruegge. “Interactive learning: Increasing student participation through shorter exercise cycles.” In: *19th Australasian Computing Education Conference.* ACM. 2017, pp. 17–26.

[Kuh96] Thomas S. Kuhn. *The Structure of Scientific Revolutions.* University of Chicago Press, 1996. ISBN: 0226458083.

[LY17] Yang Li and Tao Yang. “Word Embedding for Understanding Natural Language: A Survey.” In: *Studies in Big Data.* Springer International Publishing, May 2017, pp. 83–104. DOI: `10.1007/978-3-319-53817-4_4`.

[Lik32] Rensis Likert. “A Technique for the Measurement of Attitudes.” In: *Archives of Psychology* 22.140 (1932), pp. 1–55.

[LC04a] Lucian Vlad Lita and Jaime Carbonell. “Instance-Based Question Answering: A Data-Driven Approach.” In: *2004 Conference on Empirical Methods in Natural Language Processing.* EMNLP-2004. Barcelona, Spain: Association for Computational Linguistics (ACL), July 2004, pp. 396–403. URL: `https://aclanthology.org/W04-3251`.

[LC04b] Lucian Vlad Lita and Jaime Carbonell. “Unsupervised Question Answering Data Acquisition from Local Corpora.” In: *13th ACM International Conference on Information and Knowledge Management.* CIKM ’04. Washington, D.C., USA: Association for Computing Machinery (ACM), 2004, pp. 607–614. ISBN: 1581138741. DOI: `10.1145/1031171.1031283`.

[LWZ06] Chuanhan Liu, Yongcheng Wang, and Fei Zheng. “Automatic Text Summarization for Dialogue Style.” In: *2006 IEEE International Conference on Information Acquisition.* IEEE, 2006. DOI: `10.1109/icia.2006.306009`.

[Llo82] Stuart P. Lloyd. “Least squares quantization in PCM.” In: *IEEE Transactions on Information Theory* 28.2 (March 1982), pp. 129–137. DOI: `10.1109/tit.1982.1056489`.

[Mac67] James B. MacQueen. “Classification and analysis of multivariate observations.” In: *5th Berkeley Symposium on Mathematical Statistics and Probability.* Vol. 1: Statistics. 1967, pp. 281–297.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval.* Cambridge University Press, July 2008. DOI: 10.1017/cbo9780511809071.

[Mar03] Steven P. Marrone. "Medieval philosophy in context." In: *The Cambridge Companion to Medieval Philosophy.* Cambridge University Press, August 2003, pp. 10–50. DOI: 10.1017/ccol0521806038.002.

[May+09] Richard E. Mayer, Andrew Stull, Krista DeLeeuw, Kevin Almeroth, Bruce Bimber, Dorothy Chun, Monica Bulger, Julie Campbell, Allan Knight, and Hangjin Zhang. "Clickers in college classrooms: Fostering learning with questioning methods in large lecture classes." In: *Contemporary Educational Psychology* 34.1 (January 2009), pp. 51–57. DOI: 10.1016/j.cedpsych.2008.04.002.

[MH17] Leland McInnes and John Healy. "Accelerated Hierarchical Density Based Clustering." In: *2017 IEEE International Conference on Data Mining Workshops.* ICDMW '17. IEEE, November 2017. DOI: 10.1109/icdmw.2017.12.

[MHA17] Leland McInnes, John Healy, and Steve Astels. "hdbscan: Hierarchical density based clustering." In: *The Journal of Open Source Software* 2.11 (March 2017), p. 205. DOI: 10.21105/joss.00205.

[MW06] Olena Medelyan and Ian H. Witten. "Thesaurus based automatic keyphrase indexing." In: *6th ACM/IEEE-CS joint conference on Digital libraries.* JCDL '06. Association for Computing Machinery (ACM), 2006. DOI: 10.1145/1141753.1141819.

[Mik+13] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient Estimation of Word Representations in Vector Space." In: *1st International Conference on Learning Representations. Workshop Track.* ICLR '13. Scottsdale, Arizona, USA, May 2013. DOI: 10.48550/arXiv.1301.3781.

[Mis+11] Hemant Misra, François Yvon, Olivier Cappé, and Joemon Jose. "Text segmentation: A topic modeling perspective." In: *Information Processing & Management* 47.4 (July 2011), pp. 528–544. DOI: 10.1016/j.ipm.2010.11.008.

[Mis+09] Hemant Misra, François Yvon, Joemon M. Jose, and Olivier Cappe. "Text Segmentation via Topic Modeling: An Analytical Study." In: *18th ACM Conference on Information and Knowledge Management.* CIKM '09. Hong Kong, China: Association for Computing Machinery (ACM), 2009, pp. 1553–1556. ISBN: 9781605585123. DOI: 10.1145/1645953.1646170.

[Mit+99] Vibhu O. Mittal, Mark Kantrowitz, Jade Goldstein, and Jaime G. Carbonell. "Selecting Text Spans for Document Summaries: Heuristics and Metrics." In: *16th National Conference on Artificial Intelligence and 11th Conference on Innovative Applications of Artificial Intelligence*. Orlando, Florida, USA: AAAI Press / The MIT Press, July 1999, pp. 467–473. URL: `http://www.aaai.org/Library/AAAI/1999/aaai99-067.php`.

[Ng+19] Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. "Facebook FAIR's WMT19 News Translation Task Submission." In: *4th Conference on Machine Translation*. Vol. 2: Shared Task Papers. Florence, Italy: Association for Computational Linguistics (ACL), August 2019, pp. 314–319. DOI: `10.18653/v1/W19-5333`.

[NM06] David J. Nicol and Debra Macfarlane-Dick. "Formative assessment and self-regulated learning: a model and seven principles of good feedback practice." In: *Studies in Higher Education* 31.2 (April 2006), pp. 199–218. DOI: `10.1080/03075070600572090`.

[Nyb+02] Eric Nyberg et al. "The JAVELIN Question-Answering System at TREC 2002." In: *11th Text REtrieval Conference*. TREC '02. Gaithersburg, Maryland, USA: National Institute of Standards and Technology (NIST), November 2002. URL: `http://trec.nist.gov/pubs/trec11/papers/cmu.javelin.pdf`.

[Nyb+03] Eric Nyberg et al. "The JAVELIN Question-Answering System at TREC 2003: A Multi-Strategh Approach with Dynamic Planning." In: *12th Text REtrieval Conference*. TREC '03. Gaithersburg, Maryland, USA: National Institute of Standards and Technology (NIST), November 2003. URL: `http://trec.nist.gov/pubs/trec12/papers/cmu.javelin.qa.pdf`.

[Ott+19] Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. "fairseq: A Fast, Extensible Toolkit for Sequence Modeling." In: *2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. NAACL. Minneapolis, Minnesota: Association for Computational Linguistics (ACL), June 2019, pp. 48–53. DOI: `10.18653/v1/N19-4009`.

[PT17] Irina Pak and Phoey Lee Teh. "Text Segmentation Techniques: A Critical Review." In: *Innovative Computing, Optimization and Its Applications*. Ed. by Ivan Zelinka, Pandian Vasant, Vo Hoang Duy, and Tran Trong Dao. Springer International Publishing, November 2017, pp. 167–181. DOI: `10.1007/978-3-319-66984-7_10`.

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher Manning. "GloVe: Global Vectors for Word Representation." In: *2014 Conference on Empirical Methods in Natural Language Processing*. EMNLP. Doha, Qatar: Association for Computational Linguistics (ACL), October 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://aclanthology.org/D14-1162.

[Pet+18] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. "Deep Contextualized Word Representations." In: *2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (NAACL). Vol. 1 (Long Papers). Association for Computational Linguistics (ACL), 2018. DOI: 10.18653/v1/n18-1202.

[Pop34] Karl Raimund Popper. *Logik der Forschung – Zur Erkenntnistheorie der modernen Naturwissenschaft*. Springer, 1934. ISBN: 978-3-7091-2021-7.

[Pop59] Karl Raimund Popper. *The Logic of Scientific Discovery*. Hutchinson, 1959. ISBN: 978-0-0911-1720-7.

[PHM11] Margaret Price, Karen Handley, and Jill Millar. "Feedback: focusing attention on engagement." In: *Studies in Higher Education* 36.8 (December 2011), pp. 879–896. DOI: 10.1080/03075079.2010.483513.

[Ram03] Juan Ramos. "Using TF-IDF to Determine Word Relevance in Document Queries." In: *1st Instructional Conference on Machine Learning*. ICML '03. 2003.

[Rey99] Jeffrey C. Reynar. "Statistical Models for Topic Segmentation." In: *37th Annual Meeting of the Association for Computational Linguistics*. College Park, Maryland, USA: Association for Computational Linguistics (ACL), June 1999, pp. 357–364. DOI: 10.3115/1034678.1034735.

[RB12a] Martin Riedl and Chris Biemann. "Sweeping through the Topic Space: Bad luck? Roll again!" In: *PJoint Workshop on Unsupervised and Semi-Supervised Learning in NLP*. Avignon, France: Association for Computational Linguistics (ACL), April 2012, pp. 19–27. URL: https://aclanthology.org/W12-0703.

[RB12b] Martin Riedl and Chris Biemann. "TopicTiling: A Text Segmentation Algorithm based on LDA." In: *ACL 2012 Student Research Workshop*. Jeju Island, Korea: Association for Computational Linguistics

(ACL), July 2012, pp. 37–42. URL: https://aclanthology.org/W12-3307.

[Rij77] Cornelis Joost van Rijsbergen. “A Theoretical Basis for the Use of Co-Occurrence Data in Information Retrieval.” In: *Journal of Documentation* 33.2 (February 1977), pp. 106–119. DOI: 10.1108/eb026637.

[Rod12] C. Osvaldo Rodriguez. “MOOCs and the AI-Stanford Like Courses: Two Successful and Distinct Course Formats for Massive Open Online Courses.” In: *European Journal of Open, Distance and E-Learning* 2012/II (July 2012). URL: https://old.eurodl.org/materials/contrib/2012/Rodriguez.pdf.

[RHW86a] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning Internal Representations by Error Propagation.” In: *Parallel Distributed Processing.* Ed. by James L. McClelland and David E. Rumelhart. Cambridge: MIT Press, 1986, pp. 318–362. ISBN: 0-262-18120-7.

[RHW86b] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. “Learning representations by back-propagating errors.” In: *Nature* 323.6088 (October 1986), pp. 533–536. DOI: 10.1038/323533a0.

[ST69] Roger C. Schank and Larry Tesler. “A Conceptual Dependency Parser for Natural Language.” In: *International Conference on Computational Linguistics.* COLING. Sånga Säby, Sweden: Association for Computational Linguistics (ACL), September 1969. URL: https://aclanthology.org/C69-0201.

[Sch15] Jürgen Schmidhuber. “Deep learning in neural networks: An overview.” In: *Neural Networks* 61 (January 2015), pp. 85–117. DOI: 10.1016/j.neunet.2014.09.003.

[Sev12] Charles Severance. “Teaching the World: Daphne Koller and Coursera.” In: *Computer* 45.8 (August 2012), pp. 8–9. DOI: 10.1109/mc.2012.278.

[Sey82] Oskar Seyffert. *Lexikon der klassischen Altertumskunde: Kulturgeschichte der Griechen und Römer. Mythologie und Religion, Litteratur, Kunst und Altertümer des Staats- und Privatlebens.* Leibzig, Germany: Verlag des Bibliographischen Instituts, 1882. URL: https://hdl.handle.net/2027/hvd.32044081356115.

[Sey91] Oskar Seyffert. *A Dictionary of Classical Antiquities: Mythology, Religion, Literature & Art.* London, UK: Swan Sonnenschein and Co., 1891. URL: https://hdl.handle.net/2027/gri.ark:/13960/t85j0762m.

[Sib73] Robin Sibson. “SLINK: An optimally efficient algorithm for the single-link cluster method.” In: *The Computer Journal* 16.1 (January 1973), pp. 30–34. DOI: 10.1093/comjnl/16.1.30.

[SHS10] Raheel Siddiqi, Christopher J Harrison, and Rosheena Siddiqi. “Improving teaching and learning through automated short-answer marking.” In: *IEEE Transactions on Learning Technologies* 3.3 (2010), pp. 237–249.

[Sim88] Herbert A. Simon. “The Science of Design: Creating the Artificial.” In: *Design Issues* 4.1/2 (1988), p. 67. DOI: 10.2307/1511391.

[SS09] Jana Z Sukkarieh and Svetlana Stoyanchev. “Automating Model Building in c-rater.” In: *2009 Workshop on Applied Textual Inference*. Association for Computational Linguistics (ACL), 2009, pp. 61–69.

[Try39] Robert Choate Tryon. *Cluster analysis*. Edwards Brothers, Inc., 1939. URL: https://hdl.handle.net/2027/mdp.39015005016475.

[Tu+18] Yuwei Tu, Ying Xiong, Weiyu Chen, and Christopher Brinton. “A Domain-Independent Text Segmentation Method for Educational Course Content.” In: *2018 IEEE International Conference on Data Mining Workshops*. ICDMW. IEEE, November 2018. DOI: 10.1109/icdmw.2018.00053.

[Tur50] Alan Mathison Turing. “Computing Machinery and Intelligence.” In: *Mind* LIX.236 (October 1950), pp. 433–460. DOI: 10.1093/mind/lix.236.433.

[Wer81] Paul J. Werbos. “Applications of Advances in Nonlinear Sensitivity Analysis.” In: *10th IFIP Conference*. New York City, USA, 1981, pp. 762–770.

[Wie14] Roel J. Wieringa. *Design Science Methodology for Information Systems and Software Engineering*. Springer, 2014. ISBN: 978-3-662-43838-1. DOI: 10.1007/978-3-662-43839-8.

[Wu+07] Yun Wu, Yan Zhang, Si-ming Luo, and Xiao-jie Wang. “Comprehensive Information Based Semantic Orientation Identification.” In: *2007 International Conference on Natural Language Processing and Knowledge Engineering*. IEEE, August 2007. DOI: 10.1109/nlpke.2007.4368043.

[Yaa97] Yaakov Yaari. “Segmentation of Expository Texts by Hierarchical Agglomerative Clustering.” In: *International Conference Recent Advances in NLP*. RANLP ’97. Tzigov Chark, Bulgaria, September 1997.

[Zub38] Joseph Zubin. “A technique for measuring like-mindedness.” In: *Journal of Abnormal and Social Psychology* 33.4 (October 1938), pp. 508–516. DOI: `10.1037/h0055441`.