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.

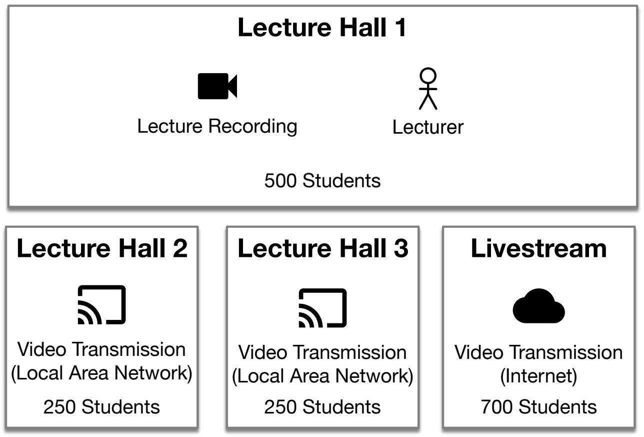


Figure 1: Mixed on-campus and virtual classroom setup employed in the summer semester 2019 at TUM for the "Introduction to Software Engineering" course.

[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.

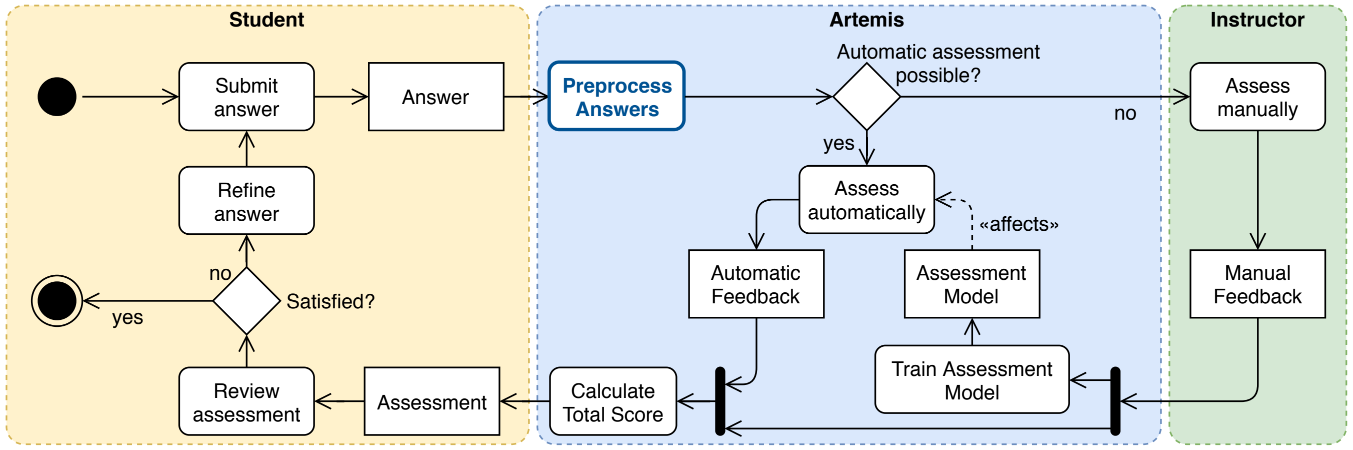


Figure 2: Workflow of the automatic assessment system for textual exercises. The "Preprocessing Answers" activity (Figure 4) includes the algorithm presented in this paper. UML activity diagram based on Bernius and Bruegge [2].

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].

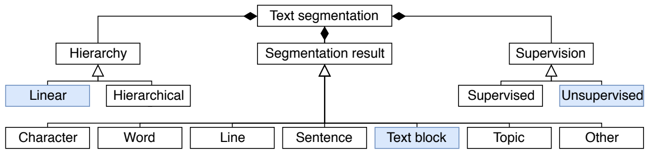


Figure 3: Taxonomy for text segmentation adapted from Pak and Teh [34]. Text segmentation types relevant for this paper are highlighted in blue color.

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].

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

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

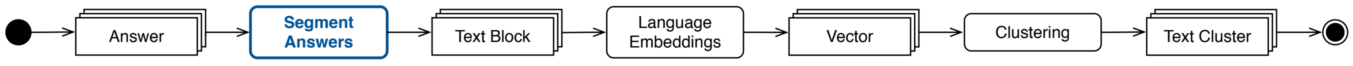


Figure 4: A detailed view into the "Preprocess Answers" activity (Figure 2) performed by the assessment system before the grading, depicted using a UML activity diagram.

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.

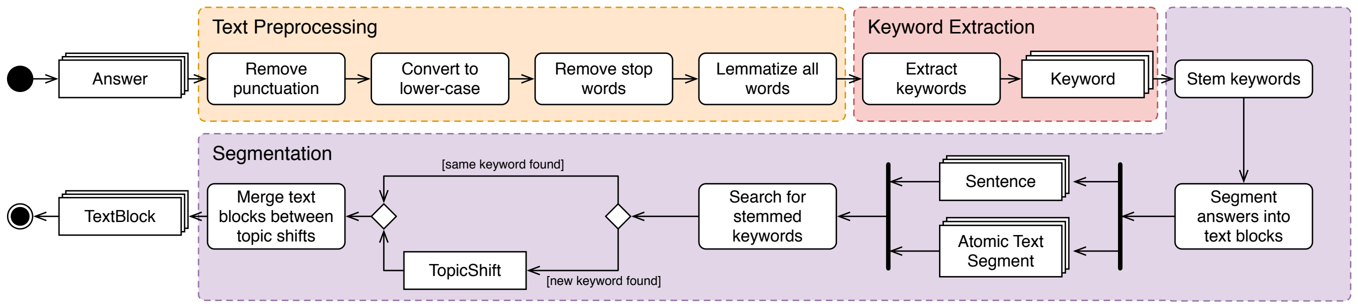


Figure 5: The segmentation algorithms flow of events depicted using a UML activity diagram based on Bernius et al. [3].

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].

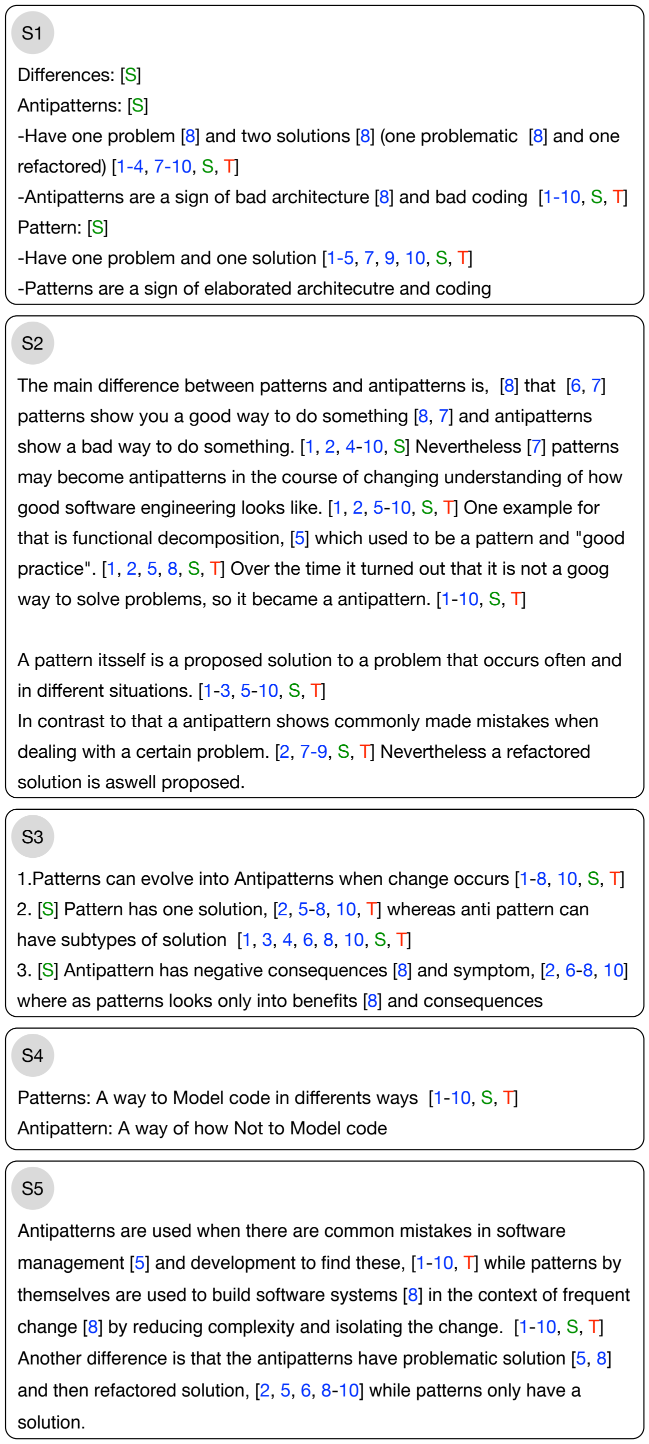
Submission	Topic Modeling		Syntactical Separators	
	Recall, %	Precision, %	Recall, %	Precision, %
S1	100	100	100	50
S2	75	60	100	67
S3	75	100	50	50
S4	100	100	100	100
S5	67	100	30	100
Average	83.4	92	76	73.4

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].

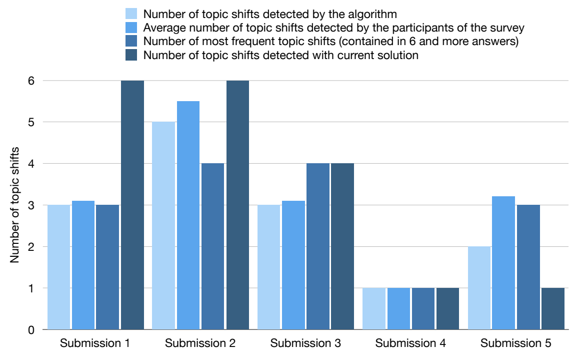


Figure 7: Comparison of the number of detected topic shifts by the current and proposed systems as well as the participants.

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.

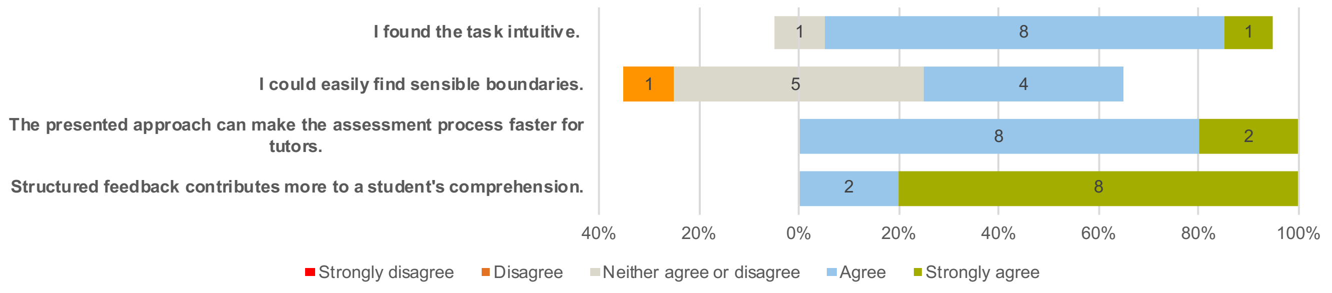


Figure 8: Participants response on their subjective impression of the approach ranked on a five-point scale based on Likert [27]. (n = 10)

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