International Journal of Artificial Intelligence in Education (2024) 34:247–273
https://doi.org/10.1007/s40593-022-00322-1

ARTICLE

Reducing Workload in Short Answer Grading Using Machine Learning

Rebecka Weegar[1] · **Peter Idestam-Almquist**[1]

Accepted: 25 October 2022 / Published online: 28 February 2023

Abstract

Machine learning methods can be used to reduce the manual workload in exam grading, making it possible for teachers to spend more time on other tasks. However, when it comes to grading exams, fully eliminating manual work is not yet possible even with very accurate automated grading, as any grading mistakes could have significant consequences for the students. Here, the evaluation of an automated grading approach is therefore extended from measuring workload in relation to the accuracy of automated grading, to also measuring the overall workload required to correctly grade a full exam, with and without the support of machine learning. The evaluation was performed during an introductory computer science course with over 400 students. The exam consisted of 64 questions with relatively short answers and a two-step approach for automated grading was applied. First, a subset of answers to the exam questions was manually graded and next used as training data for machine learning models classifying the remaining answers. A number of different strategies for how to select which answers to include in the training data were evaluated. The time spent on different grading actions was measured along with the reduction of effort using clustering of answers and automated scoring. Compared to fully manual grading, the overall reduction of workload was substantial—between 64% and 74%—even with a complete manual review of all classifier output to ensure a fair grading.

Keywords Short answer grading · Automatic grading · Machine learning · Cluster based sampling

✉ Rebecka Weegar
rebecka.weegar@umu.se

Peter Idestam-Almquist
pi@dsv.su.se

[1] Department of Computer and Systems Sciences, DSV, Stockholm University, Stockholm, Sweden

Introduction

Exam grading is a time-consuming and repetitive task, where automated grading methods have been proposed as a complement to manual grading, often motivated by its potential to reduce workload for teachers. Automated grading and assessment have been investigated in diverse domains, such as automated assessment of programming exercises (Souza et al., 2016), essay grading (Ke & Ng, 2019), and short answer grading (Burrows et al., 2015).

Here, we evaluated how automated short answer grading can reduce workload in exam grading during an introductory computer science course. This course is one of the courses taught during the first term for computer science students. The exam covers a wide set of topics related to computer science and the answers to most of the questions are relatively short, up to a couple of sentences. To get a passing grade on this course, a student has to complete a set of programming exercises, and the result on the exam fully determines the students' grade in the course.

This work had two main aims. First, to investigate the possibility for introducing machine learning support for grading in existing courses, where grading previously has been performed manually. Our second aim was to evaluate the potential for reducing the manual workload associated with grading. The work described here is intended to be applicable for courses on other topics, with similar examination procedures, for comparatively large class sizes and to be relatively simple to implement and use.

Stakeholders in Automated Grading

Automated scoring often make use of machine learning methods and natural language processing (NLP) techniques to assign scores to student answers, see Section 2. While accurate NLP and machine learning tools are important for automated grading, there are several stakeholders to consider when developing automated graders (Madnani & Cahill, 2018). It is also crucial to examine automated grading in the setting in which it is intended to be used. Both students and teachers need to be able to trust a grading system, and the consequences for students, in particular, can be severe if an automated grader is not accurate and fair. It is therefore not only classifier performance that is relevant for automated grading, but also the effects of using automated methods on the overall grading process.

When machine learning is used for a specific goal, such as assigning scores or grades to exam answers, some degree of manual work is typically required. This manual work can, for example, consist of labeling training data for a classifier. This means that machine learning does not eliminate the need for manual work in exam grading. Rather, the workload is divided into manual work and work performed by the machine learning algorithms. This division comes with a trade-off between the accuracy or correctness of the grader and the effort required during the creation of the grader. The goal of this work is therefore to evaluate how machine learning can reduce manual workload in grading while retaining a correct grading of high quality. Our work focuses both on the teacher, in terms of reducing workload through

machine learning support, and on the student, by ensuring that all scoring decisions are reviewed for correctness.

Quality in Automated Grading

It is crucial that students get a correct and fair assessment of their work, regardless of whether the grading is performed manually or if it is assisted by machine learning. To evaluate if an automated grader provides correct assessments, all common metrics for machine learning based classifiers, such as precision, recall, f-score, accuracy, and AUC, have been used in previous works.

With a trade-off between correctness in grading and the amount of manual work required, it would be helpful to set a threshold for an acceptable grading quality. One possibility for setting such a threshold is using inter-annotator agreement. By letting two or more expert graders grade the same answers, and calculating inter-annotator agreement scores, such as Cohen's kappa (Cohen, 1968), the level of agreement can be estimated. It can be argued, that a machine learning based grader achieving the same agreement with an expert grader as two expert graders do with each other can be as good or useful as an expert grader.

While both of these evaluation methods—classifier evaluation and agreement scores—are valid for comparing different machine learning-based graders, they do not correspond to a guarantee that future grading will remain at the same level of quality. Such scores are also only available after-the-fact; to be able to determine if a classifier is accurate enough requires a review of its performance each time it is applied to new questions and answers.

Additionally, while it is possible for two human graders to disagree on what is the correct grade or score for an answer, they likely have valid reasons for their different assessments. If, however, an automatically assigned grade differs from a manually assigned grade, the reason for the disagreement might be spurious and not correspond to a fair assessment. It has also been shown that automatic graders can be sensitive to manipulation (Filighera et al., 2020).

For these reasons, it is the case, that even with a very accurate classifier, it is not (yet) possible to fully replace manual grading with grading based on machine learning. The output of an automated grader will still require manual review to ensure fair and correct grading, particularly in high-stakes exams.

In this study, we therefore evaluated the possibility of reducing manual workload in short answer grading using machine learning, while also reviewing the output of the automated grader. A text classifier was trained for each question on the exam, and clustering was used both to facilitate the initial grading of training data and the review of the output of the classification. The main contribution lies in the evaluation of the grading. Instead of measuring the potential for reduced work with some acceptance for misclassification, we evaluated the amount of work required, not just for the initial classification, but also for reviewing all output from the classifier, to ensure a trustworthy and accurate grading. We also propose a modular and reusable grading setup that can be used in a course setting without the need for specifically developed interfaces for grading.

Previous Work

Automated Grading

Research on short answer grading has often focused on the evaluation of machine learning and models based on natural language processing (NLP) in terms of classifier performance (Madnani & Cahill, 2018).

An early example of an NLP system developed for short answer grading is C-rater (Leacock & Chodorow, 2003). The idea behind C-rater is that correct answers can be regarded as paraphrases of a model answer, and C-rater uses several language processing methods to map student answers to these model answers. The score assigned to a single student answer is based on how well the student answer and model answer align.

Grading based on the similarity between student answers and one or more example answers has also been investigated by Marvaniya et al. (2018) and Mohler et al. (2011).

Others have used various supervised machine learning methods for text classification to assign scores to student answers (Nielsen et al., 2008; Galhardi et al., 2020; Gomaa & Fahmy, 2014).

More recently, deep learning methods have also been applied for automatic grading of short texts using different architectures such as LSTMs and transformers (Bonthu, 2021; Riordan et al., 2017; Sung et al., 2019), sometimes in combination with data augmentation methods (Kumar et al., 2017; Lun et al., 2020), often reaching improved scoring capabilities.

Reducing Manual Workload in Grading

Previous works on automated grading with the aim to reduce workload for teachers have employed different methods to reach this goal. One such approach for facilitating manual grading is clustering. Clustering algorithms have been used to find groups of similar answers which can save time and effort as it becomes possible to provide scores or feedback to groups of students simultaneously.

Evaluation has indeed shown that clustering can decrease the amount of manual work associated with grading. One example of using clustering for short answer grading is the work by Basu et al. (2013), where a similarity score for short text answers was learned and used to find similar answers with hierarchical clustering. They found a large reduction in workload measured as the number of grading actions. In a subsequent article, their approach for clustering was integrated into a custom grading interface and evaluated further. Clustering was found to decrease the time spent on grading, and the teachers who took part in the evaluation preferred cluster-based grading (Brooks et al., 2014).

Horbach et al. (2014) investigated the trade-off between workload and correctness in grading using clustering by setting thresholds for grading accuracy. They

found it possible to grade short answers for German as a second language with between 85% and 90% accuracy while only manually grading 40% of the answers.

Horbach and Pinkal (2018) suggested using semi-supervised clustering for short answer grading. This was achieved by imposing constraints on the clustering by selecting pairs of answers that should not be clustered together. The clusters were graded by scoring the answer closest to the centroid of the cluster, assuming that this answer was representative for the cluster, and propagating the score to the remaining answers in the cluster. Evaluation using quadratically weighted Cohen's kappa found that the semi-supervised clustering produced more accurate clusters.

An alternative for reducing workload in automated grading is active learning. Active learning is relevant when a classifier is trained to score answers and can reduce the amount of labeled data needed for training the classifier. A small initial set of instances are selected and manually labeled and a machine learning model is trained on this initial set. Additional instances are added to the training data iteratively until some criterion is met and each round of training is followed by a selection of new instances to label. These can, for example, be selected using uncertainty sampling, i.e. selecting the instances the machine learning model has the least confidence in classifying (Settles, 2009).

Horbach and Pal (2016) evaluated several modes of active learning for short answer grading with uncertainty sampling yielding the best results in terms of linearly weighted Cohen's kappa. Kishaan et al. (2020) measured reduced effort by logging the number of clicks required for grading and compared this to the accuracy of the scorings.

Geigle et al. (2016) also discussed how to organize cooperation between human graders and machine learning for automated grading and the trade-off between quality and effort. They proposed using active learning to reduce the effort needed to create an automated classifier and to rank answers instead of scoring them, leaving it to the instructor to stratify the ranked assignments manually. They argued that less effort is required for creating training data for ranking, as it could be easier to determine, for a pair of answers, which one has the higher quality, compared to scoring each answer.

In addition to grading clusters or using active learning, reducing effort by focusing manual work on a subset of answers which are more challenging to automatically grade has also been evaluated. One example is the work by Kulkarni et al. (2014), where the graders were not teachers, but peers. First, classification was performed, and the classifier's confidence for individual answers guided how many peers next should grade each answer. They reported the accuracy of the grading together with the effort required for achieving that accuracy, and also the students' trust in the system.

A related method was described by Mieskes & Pado (2018). They evaluated strategies for prioritizing more challenging answers, including using an ensemble of classifiers to first grade the answers, and next manually reviewing the answers if the classifiers disagreed on the scoring. They reported a large reduction in grading effort in terms of precision and recall for the scoring and used Fleiss' kappa as a measure of agreement.

The demands on an automated grader depend on the setting in which it is employed. Often, however, the evaluation of automated graders is carried out on publicly available data sets, rather than during actual grading (Azad et al., 2020), partially due to the risk of inaccuracies in grading. Azad et al. (2020) evaluated a slightly different approach for handling inaccuracies. The grading was carried out interactively and the students got immediate feedback on their answers. The students were allowed to reattempt answers if the answers were not initially approved, and could also appeal for reevaluation if they did not agree with the final grading. This setup was evaluated during a "high-stakes exam", where the exam grade corresponded to 10% of the final grade on the course. They reported accuracy, AUC scores, and F1 scores for the classification, and also evaluated the students' perception of and trust in the system, which were found to be overall high.

These related works on reducing manual workload in grading emphasize the importance of considering how to best combine automated, machine learning based grading with manual effort. Previous works show several possible paths to get closer to the goal of automated grading, but also unsolved challenges for using automated or semi-automated grading in practice, both in terms of the accuracy of automated graders and when considering overall fairness from a student perspective.

Methods

Data Set

The questions and answers used in this study come from an exam in an introductory course to computer science consisting of 64 questions. In total, 438 students took part in the exam and each student was asked 36 of the 64 questions. Empty answers were removed from the data set as these could be directly assigned a zero score and would not contribute to the evaluation of the automated grading. After removing empty answers, each question had on average 230 answers. The exam answers were mainly written in Swedish, with some use of English as the students were allowed to answer in either Swedish or English. This set of answers was used to evaluate the overall approach for machine learning supported exam grading and for estimating the reduction in workload. Additionally, example solutions written by the teachers were provided to the graders during grading. The example solutions consisted of a single, correct answer to each question and were used for guidance during grading. The example solutions were also made available to the students after the grading was finished.

In addition to this exam, a retake exam from the same course was used to measure the time spent on different grading activities. The retake exam, which was open both to students who had failed or missed the original exam, had fewer participants, 73 students, but was otherwise similar in structure and contents to the ordinary exam.

In previous years, the exam has been graded by two teachers on the course who formulated and graded about half of the questions each. This approach was followed during the evaluation of the machine learning supported grading for both the ordinary and the retake exams.

Question and Answer Types

The exam covers several topics such as programming languages, networking and the Internet, and data abstractions and manipulations. A few different types of questions were included in the exam. Many of the questions ask for definitions or specific concepts, such as the question: "Name the four Internet software layers", and some of the questions required the students to compare different methods or techniques or perform calculations. The list below gives examples of two questions with three answers each, where the student's answers have been translated into English when necessary:

- **Name the four Internet software layers**
  - HTTP. HTTPS. SMTP. FTP
  - Transport layer, link layer, application layer, network layer
  - Application layer, Transport layer, Network layer & Link layer
- **Why is the halting problem interesting from a computational theoretical perspective?**
  - It is interesting because not all programs are terminating.
  - The halting problem is unsolvable which shows that there are problems that can not be solved using algorithms
  - The halting problem is not solvable, and it is a proof of the existence of problems that cannot be solved with the help of computers

The type of knowledge and abilities tested in the exam can be described using the knowledge dimension and the cognitive process dimension of Bloom's revised taxonomy, a framework for categorizing learning objectives (Anderson & Bloom, 2001).

The knowledge dimension in this framework includes four knowledge types: factual, conceptual, procedural, and metacognitive knowledge, where the first three knowledge aspects were relevant for this exam. Factual knowledge is the basic, necessary knowledge of course content, such as terminology. Factual knowledge is followed by conceptual knowledge, the ability to relate basic concepts to each other, and procedural knowledge, knowledge of how select and use relevant methods and techniques. The cognitive process dimension comprises five categories: remembering, applying, analyzing, evaluating, and creating. The cognitive and the knowledge dimensions can be combined to describe a learning objective. One example of such a learning objective would be "remembering factual knowledge", for example by correctly naming the protocol used for sending emails. Other combinations relevant for the exam are remembering and understanding conceptual and procedural knowledge and also applying procedural knowledge, for example when performing calculations (Anderson & Bloom, 2001).

Workload in Exam Grading

Figure 1 provides an overview of the approach for combining manual and machine learning-based grading used in this study. All answers were first clustered and a set

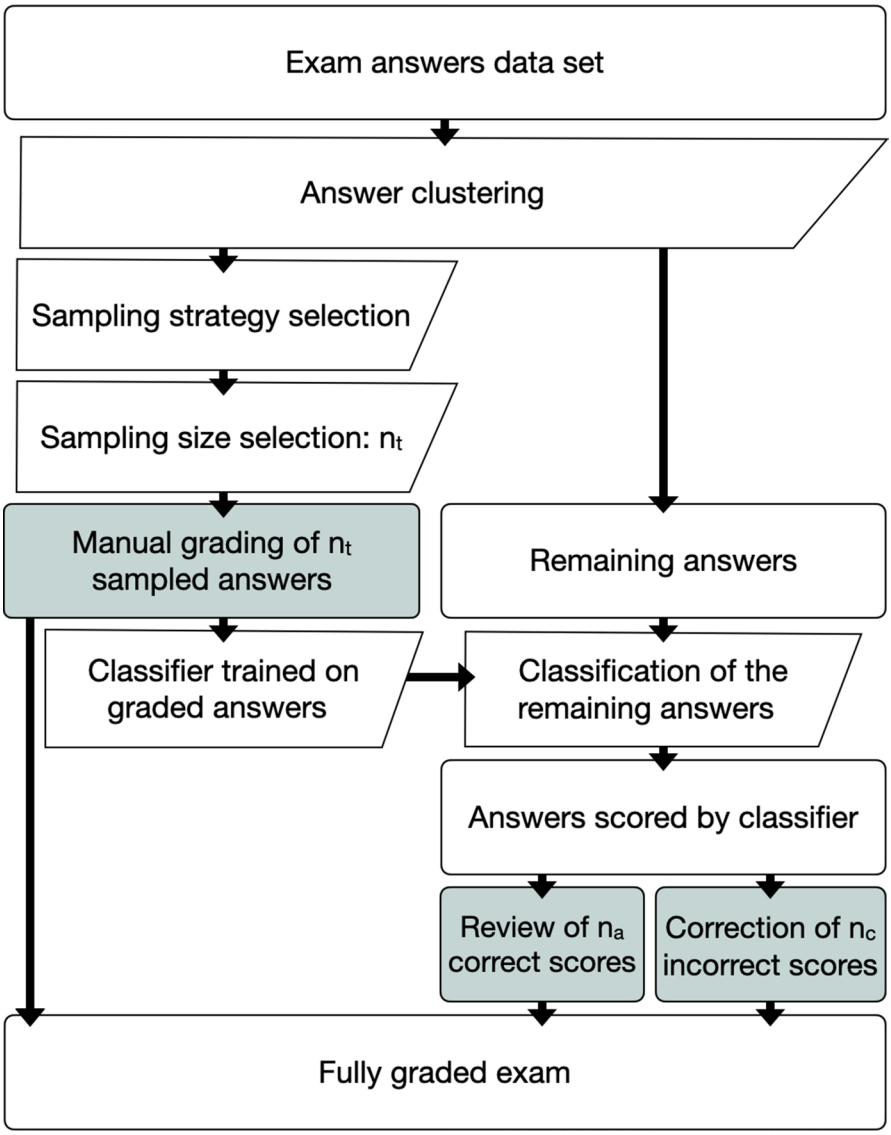


Fig. 1 Overview of the process for exam grading. Shaded boxes represent manual work. As a first step, student answers were clustered into groups of similar answers. A subset of these answers was sampled using one of the sampling strategies described in Section 3.4. The sampled answers were graded and used as training data and the trained classifiers scored the remaining answers in the data set. The scores were reviewed by the graders and corrected if incorrect

of answers were sampled for each question. These answers were manually graded and used as training data for a classifier. Different sampling sizes were evaluated, see Section 3.5. The classifier was next used to score the remaining answers. The output of the classifier was manually reviewed and corrected when needed. During both of these phases, grading and reviewing, the answers were presented in clusters so that similar answers could be assessed simultaneously and a spreadsheet program was used to display the answers and assign or review scores.

With this approach, two types of manual work are required in addition to the work performed by machine learning. First, manual grading of answers used as training data, and second, review of the output of the classification algorithms. With an exam consisting of N answers, n_t is the number of answers manually graded for training. The remaining answers are graded by the classifier, where one part, n_a, will be automatically graded and correct, and the remaining answers n_c will require manual correction. This gives that $N = n_t + n_a + n_c$.

Each answer, also those assigned a correct score by the classifier, will require some manual review, but it is also true that different amounts of effort are associated with each of the manual grading tasks. The effort in manually grading one answer for training is λ_t; for reviewing one correct answer, the effort is λ_a and, for correcting a mistake made by the classifier, the effort is λ_c. The complete workload for grading one exam can therefore be said to be: $\lambda_t n_t + \lambda_a n_a + \lambda_c n_c$

Clustering and classification can both contribute to the reduction of time spent on grading. By clustering the answers, λ_t, λ_a and λ_c are all reduced compared to grading an exam without first clustering the answers since a decision made about a grade for one answer often can be transferred to other answers in the same cluster. The automatic grading further reduces λ_a, as it requires less effort to confirm the score suggested by the classifier compared to manually grading an answer. It can therefore be assumed that $\lambda_t > \lambda_a$ and $\lambda_c > \lambda_a$. The evaluation of these assumptions is described in Section 3.7 and the results of the evaluation in Section 4.1

An accurate classifier is therefore essential to reduce the workload, as it will increase the number of correctly scored answers, n_a, and thereby decrease n_c, the number of scores that needs to be corrected. To also reduce n_t, the number of answers that are graded for training the classifier, the sampling strategy is important, i.e. how to select a set of answers for manual grading in such a way that a minimal set of answers is required while still retaining the accuracy of the classifier. The n_a correctly classified answers will however always have to be reviewed, regardless of which strategy is selected. Therefore, a better sampling strategy is one that results in the minimum amount of manual work for grading and correcting, here described by $\frac{n_t + n_c}{N}$. Note that with minimal training data and maximum accuracy, the work of reviewing scored answers $\lambda_a n_a$, will still be required.

Each step of this approach is described in more detail in the following sections. Section 3.3 presents the clustering methods. Clustering was used to find groups of similar answers both to facilitate the grading of those answers, and to inform the selection of training data. The included strategies for selecting training data are presented in Section 3.4, sampling strategies, and Section 3.5 describes the different amounts of training data sampled from the available student answers. Next,

Section 3.6 provides details about the classification step in which classifiers were trained on the sampled training data and applied to the remaining answers. The final section of the methods chapter, Section 3.7 describes how the timed grading was performed.

Clustering of Answers

The answers were clustered for two purposes: to be able to display similar answers directly to the graders, and to select answers for training data. This gave two requirements for the clustering algorithm. The first requirement was the possibility to set the number of clusters manually to get a cluster size suitable for grading. The clusters should be large enough to encompass several similar answers, but small enough to provide an easy and fast overview of the included answers. The second requirement was that the clustering algorithm should give some type of measure of representativeness for each instance in a cluster to make it possible to select representative answers to include in the training data.

Two clustering algorithms fulfilling these requirements were selected, k-means and Gaussian mixture models (GMM), both implemented in Scikit-learn (Pedregosa et al., 2011). The two algorithms were evaluated using silhouette scores. The silhouette score for a clustering ranges from -1 to 1 and compares the distance between samples in the same cluster with the distance between clusters, and higher silhouette score corresponds to a better clustering. The scores for k-means were higher, with a mean silhouette of 0.42 compared to a mean silhouette of 0.40 for GMM. K-means was therefore selected for the main experiments.

All questions and example solutions were tokenized–divided into words–and lemmatized using Stanza (Qi et al., 2020). Lemmatization exchanges a word for the basic form of the word, giving different inflections of the same word the same representation. A small set of stop words were removed from the texts. Stop words are common words that have little impact on the meaning or contents of a text, such as "the" and "a". The average length of an answer in the data set was 13 tokens and after removing stop words, the average length was 11 tokens.

For the clustering, only unigram features were included, as one purpose of the clustering was to give an overview of similar answers. Textual similarity can of course be considered from different perspectives, but this decision was based on the intuition that lexical similarity between answers in the same cluster would make it easier to quickly get an overview of the content of the answers in each cluster.

The true number of clusters for a question could be considered to correspond to the number of different scores assigned for the answers to that question. This number was however not known at the time of clustering, and, as discussed by Horbach and Pinkal (2018), the number of natural clusters among a set of answers is likely larger than the number of different scores that should be assigned for those answers. For example, answers given the same partial score might have been awarded that score for different reasons and their contents might have very little overlap.

We, therefore, opted to set the number of clusters to benefit the manual grading and review, rather than trying to create a 1-to-1 mapping between scores and

clusters. Too few and large clusters would be difficult to assess, and too many clusters would reduce the time saved on grading complete clusters. An average cluster size of around ten answers per cluster was determined to be suitable, and therefore each question was clustered into 20 clusters. To evaluate the impact of the number of clusters on the overall classification performance, the clustering was also repeated three additional times with k set to 10, 30, and 40 respectively. Figure 2 shows an example of two answer clusters.

Sampling Strategies

For each question, a set of answers were manually graded and subsequently used as training data for a classifier trained to score the remaining answers. To create an accurate classifier, it is preferable to have a set of training data that is representative of all the answers in the complete data set so the classifier can learn to grade different types of answers to the same question. At the time of manual grading for creating training data, it is however not known how to best select the answers to include in the training set. Therefore, six sampling strategies were used to select which answers to grade in the first phase.

Each sampling strategy creates an ordering of all of the answers to each exam question, and from this ordering, a sample of answers was graded manually. Five of the sampling strategies were based on clustering the answers into groups of similar answers and the following five cluster-based sampling strategies were included:

Cluster order: sampling full clusters in the order given by the clustering algorithm.

Cluster size: sampling full clusters, in order from the smallest to the largest cluster.

Cluster distance: sampling full clusters by iteratively selecting clusters at the furthest distance from previously selected clusters.

Cluster centroid: sampling instances from different clusters, starting from the center of each cluster. With 20 sampled answers and 20 clusters, the center-most answer from each cluster would be included.

Random cluster instance: sampling instances by randomly selecting answers from different clusters. With 20 sampled answers and 20 clusters, 20 answers from

Fig. 2 Examples from two clusters answering the question "In object-oriented programming, you have classes and objects. In addition to this, there are three features that characterize object-oriented programming, which ones?" Answers have been translated into English

```
###  CLUSTER 1:  ###
Lists, Stacks, Boolean
Lists, strings
Lists, tuples
###  CLUSTER 2:  ###
Inheritance, encapsulation, polymorphism.
Inheritance, encapsulation and polymorphism.
1. inheritance, 2. encapsulation, 3. polymorphism
Encapsulation. Polymorphism. Inheritance.
Inheritance, encapsulation, polymorphism.
inheritance, encapsulation, polymorphism.
```

different clusters would be included, not taking the distance from the cluster center into account.

For the first three of these strategies, all answers contained in a cluster were sampled before moving on to the next cluster. For the last two strategies, cluster centroid and random cluster instance, the sampling of individual answers was distributed over the clusters.

In addition to the five cluster-based sampling strategies, **random sampling** over all of the answers without previous clustering was also evaluated. This strategy was not influenced by the clusters.

Cluster-based sampling has been used previously for selecting training data. Yen and Lee (2009) used cluster-based sampling to select instances from the majority class to be excluded during training to get a more balanced data set. In this case, however, the class distribution was not known when the instances were sampled.

Sample Sizes

An ideal sampling strategy would allow for grading only a small set of answers while still being able to train an accurate classifier on the set of graded answers. To evaluate how many instances would be required for training a useful classifier, each sampling strategy was evaluated with ten different sample sizes, from 10 to 100 answers in increments of 10.

Answer Classification

Example solutions to the questions were available during grading, and the degree of similarity between a student's answer and the example solution is likely indicative of the quality of the student's answer. Therefore, the cosine similarity of each answer and the corresponding example solution was calculated for vector representations of the answers and example solutions. The maximum value for cosine similarity is 1, which would denote identical vectors (Singhal, 2001). The vector representations were based on sentence embeddings, which is a representation of a piece of text where semantically similar texts receive similar vectors. To generate the vector representations, Sentence BERT Reimers & Gurevych (2019) was used. Sentence BERT is an extension of the seminal BERT model (Devlin et al., 2018). A multilingual pre-trained sentence BERT model was used (Reimers & Gurevych, 2020), which was preferable in our case since answers sometimes included English text as well as Swedish text. In addition to the similarity scores, the lemmas included in the answers were also used as features during the answer classification.

During the first grading phase, the answers sampled for training data were assigned a score by one of the two graders. These scores ranged from zero to one, and partial scores were given to partially correct answers. For a majority of the questions, three different scores, for example, 0, 0.5, and 1.0, were assigned, but in some cases, four different scores were given to answers to the same question. These manually graded answers were next used to train classification algorithms. Three commonly used classification algorithms were evaluated to find a suitable option:

Random Forest, a robust classifier based on an ensemble of decision trees, Support Vector Machines (SVM), which can perform well for high-dimensional data, which is often the case with text input, and Complement Naive Bayes (CNB) which is suitable for imbalanced data sets, which can be the case when only a few students get a partial score for a particular question. The Scikit learn library was used for the classification (Pedregosa et al., 2011).

5-fold cross-validation was performed to determine how to best represent the answers to each question, either using tf-idf weighting of the words in each answer or simply representing each answer with a vector of binary word occurrences. Tf-idf weighting gives more weight to words that are common in a specific answer, but rare in the whole collection of answers. In addition, the cosine similarity between the student answers and the model answers, derived from Sentence BERT embeddings, was also included as a feature. Cross-validation was also used to determine the suitable n-gram range for each answer. Based on the results of the cross-validation, the models were retrained using all manually graded answers, giving a unique model for each question. These models were next applied to the remaining answers, providing a score for each of them.

After automatically scoring the remaining answers, the scores were reviewed to ensure their correctness. During the review, the answers were displayed to the grader in the order determined by the clustering algorithm to facilitate the review of similar answers as shown in Fig. 3.

Measuring the Time Spent on Grading

The grading procedure described here was also applied to a retake exam, during which the time spent on each grading activity was timed.

CLUSTER 1	Classification	Correction	Comment
Assembler, compiler and interpreter	0.0		
compiler. interpreter.	0.0		
compiler, parser and interpreter.	0.0	0.33	
interpreter, condenser.	0.0		
interpreter, compiler	0.0		
CLUSTER 2			
Lexcial analyser. Parser. Code generator.	1.0		
Lexical analyser, Parser and Code generator.	1.0		
Lexical analyser - Parser - Code Generator .	1.0		

Fig. 3 The exam answers as they were displayed to the grader during the review of the scores assigned by the classifier. The figure shows parts of two clusters with student answers, scores, and columns for correcting the scoring and commenting answers. During the grading, colors were used to signify the classifier's confidence (a value between 0 and 1). Of the three colors displayed here, green corresponded to a confidence of ≥ 0.8 but below 0.9, and blue to a confidence ≥ 0.9. The corrected score was marked yellow, meaning that the confidence in this score was lower than the other answers shown here (between 0.5 and 0.7). The answers have been translated into English

The purpose of timing the grading was to investigate if, and to what degree, machine learning based grading support could contribute to reducing workload in exam grading. Both types of support were evaluated, displaying answers to the grader in groups of similar answers, and displaying score suggestions for each answer to the grader. The two types of support were evaluated both individually and when used together.

The timed grading followed the steps used for grading the ordinary exam. The "cluster centroid" sampling strategy was used, meaning that the answers were clustered, and one answer from each cluster center was sampled. These answers were graded without any machine learning support. The grading was timed and this served as a baseline for the remaining steps. The graded answers were used as training data for a Random Forest classifier.

The next task for the graders was to review the output of the classifiers trained on the sampled answers. For this review step, the questions in the retake exam were divided into four groups, A-D, to estimate the time needed for grading when using the different kinds of machine learning-based support, and to evaluate the assumptions regarding workload defined in Section 3.2.

For group A, the answers were given score suggestions, but the answers were not clustered. For group B, the answers were clustered, but no score suggestions were provided. The answers in groups C and D were clustered and score suggestions were also shown to the grader. Group D corresponds to the approach described in the rest of this paper (Table 1).

For group C, the grading was performed in two steps to measure the time it took to review answers compared to the time required for correcting answers. First, each score suggestion was reviewed and incorrectly scored answers were marked. Next, the scores of the marked answers were corrected. The two steps were individually timed.

When grouping the questions into groups A, B, C, and D, we aimed to create a balanced mix of questions in the different groups. Consideration was taken both to the character of the question in relation to Bloom's taxonomy, and the different kinds of expected answers, such as results of calculations or longer free-text answers. We also aimed to separate questions on similar topics into different groups.

Table 1 The different combinations of machine learning support used during the timed experiments. Separate correction means that the review step was separated from the correction step

Group	Score suggestions	Clusters	Separate correction
A	✓		
B		✓	
C	✓	✓	✓
D	✓	✓	

Results

Time Spent on Grading

A retake exam was graded and the time required for different grading actions was timed (see Section 3.7). The results are displayed in Table 2. As the table shows, both types of support, clustering and score suggestions, reduced the time spent on grading answers when used individually for groups A and B. The reduction of time for each method was 69%. Combining clustering with score suggestions gave the largest reduction in average time spent on grading an answer: 73%. The time saved was compared group-wise, i.e. to the average time spent on grading answers in the same group without the support of machine learning.

For group C, the average time spent on grading answers without support was 8.9 seconds. For this group, the review was carried out in two steps, first, each incorrect score suggestion was marked. This step took on average 2.1 seconds for an incorrect answer, while correction took on average 17.7 seconds for a corrected score. The time for correction was influenced by one particular question where the average correction took about 43 seconds. The reason for the extra time needed for assessing the answers to this question was likely that there were many partially correct answers given to this question, where these answers were also different from each other. A clearly correct or incorrect answer is relatively easy to identify, compared to a unique and partially correct answer which requires a more thorough individual assessment.

Excluding that question, the mean time spent on correction for the remaining answers was about 8.8 seconds, i.e. similar to what it would take to manually grade an answer in this group.

In groups A, C, and D, score suggestions were provided to the grader, and in each of these groups, there were some questions with no incorrect score suggestions (four for group A and three for groups C and D). This made it possible to estimate the time spent reviewing correct score suggestions. The average time spent reviewing a correctly scored answer was 0.31 seconds for group A, 0.35 seconds for group D, and 0.27 seconds for group C.

These results can give an estimation of the individual values used to define workload in Section 3.2. The total amount of work required for grading an exam can be

Table 2 Average time spent on grading answers with different amounts of machine learning support. Time is shown in seconds. The percentage of corrections in each group is the percentage of incorrect score suggestions

Group	Grading without support	Supported grading / review	Decrease of time	Corrections
A, score suggestions	10.3 s	3.2 s	69%	7%
B, clusters	9.6 s	3.0 s	69%	-
D, clusters and score suggestions	8.8 s	2.4 s	73%	11%

described as $\lambda_t n_t + \lambda_a n_a + \lambda_c n_c$, where n is the number of answers in each of the categories: training data n_t, correctly scored answers n_a, and incorrectly scored answers n_c. Here, we found that the coefficients λ_t and λ_c were similar; the time spent on grading a single answer used for training, and the time spent on correcting a single misclassified answer was about 8-10 seconds. The time spent on reviewing correct answers, λ_a, was much smaller, with an average between 0.27 and 0.35 seconds for a single answer.

The time spent on grading was also measured for the different categories in Bloom's taxonomy. The average time for grading an answer was calculated for the knowledge dimension and the cognitive process dimension of the taxonomy.

Along the knowledge dimension, questions were categorized as requiring factual, conceptual, or procedural knowledge. The least time for grading was required for the procedural knowledge, 1.82 seconds on average. More time was required for an answer describing factual knowledge with 7.95 seconds on average, and the most for conceptual knowledge with 14.73 seconds on average.

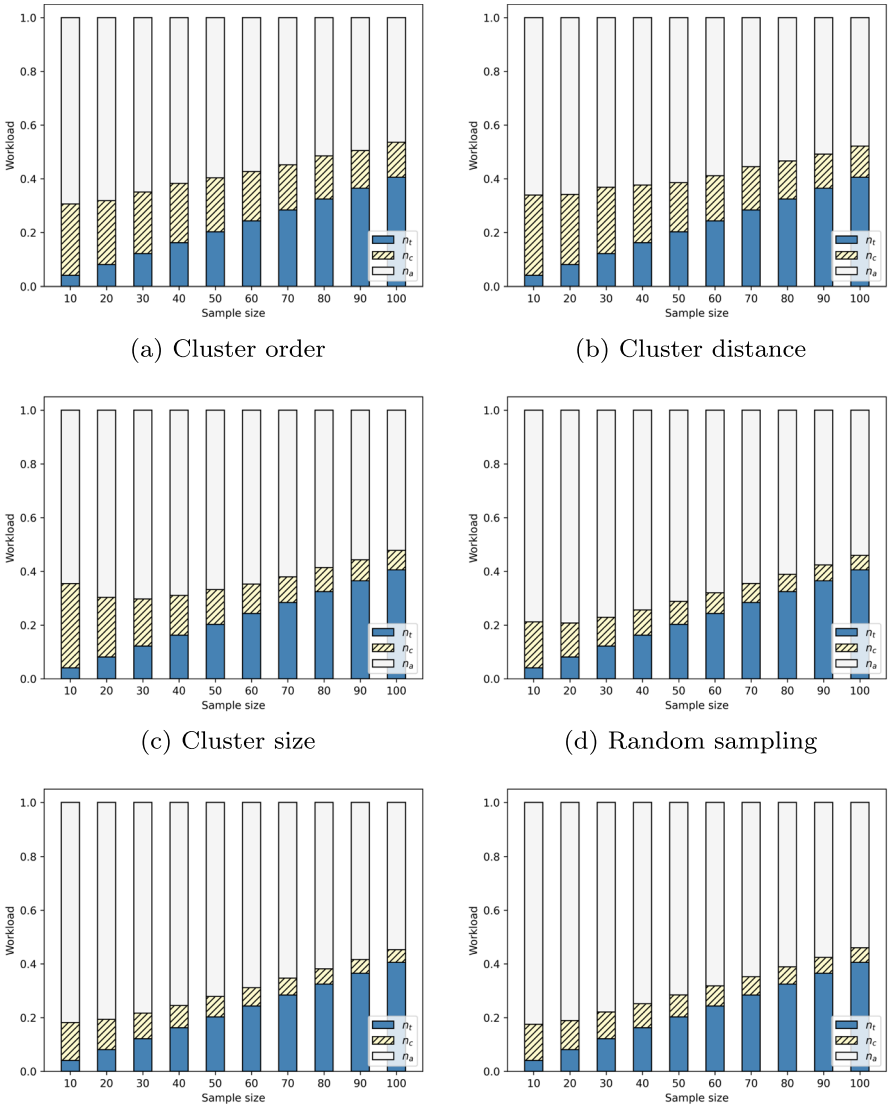
For the cognitive process dimension, the categories remembering, understanding, and applying were included. Applying was the fastest to grade, with 1.82 seconds on average (applying was only combined with procedural knowledge). This was followed by remembering with an average grading time of 10.34 seconds for an individual answer, and understanding, with an average grading time of 13.99 seconds.

With the exception of applying procedural, more high-level categories took a longer time to grade. The reason for questions where the students were tested on their ability to apply procedural knowledge being the overall fastest category to grade was likely due to these questions having a single correct answer—the result of some calculation—making it relatively easy for the grader to determine if an answer was correct or not.

Results, Sampling Strategies

To determine how to select the answers to manually grade, several sampling strategies were evaluated. The bar graphs in Fig. 4 show the distribution of workload required for grading the full exam using different sampling strategies and sample sizes. The sample size corresponds to how many answers were manually graded as training data (n_t), and the strategies were evaluated with the sample sizes 10, 20, 30, ..., 100. The share of corrected scores (n_c) is represented by the cross-hatched middle part of each bar. The accurately scored answers (n_a) are represented by the top white part of each bar. The graphs in the figure are based on the results using the Random Forest classifier.

As seen in Fig. 4, the choice of sampling strategy had a large impact on the required grading effort and the distribution of different types of work. The least manual work for labeling training data and correcting incorrect scores ($n_t + n_c$) was required for the two strategies where clustering was applied and single answers were sampled from each cluster, the cluster centroid strategy, and the random cluster instance strategy. This was followed by randomly sampling answers without



(a) Cluster order (b) Cluster distance

(c) Cluster size (d) Random sampling

(e) Random cluster instance (f) Cluster centroid

Fig. 4 Distribution of workload using different sampling strategies and sample sizes. The bottom, darker part of the bars represents n_t/N, the manually graded answers used as training data. Together with the middle, cross-hatched part, n_c/N, the corrected answers, this corresponds to the manual workload for creating training data and correcting scores. The top part, n_a, shows the correctly classified answers, which only required review. Figures were generated using Matplotlib (Hunter, 2007)

clustering. Substantially more work, in the form of grading and correcting, was required for all sampling strategies where full clusters were graded for training data.

The results in Fig. 4 are based on 20 clusters for each question. Changing the number of clusters to 10, 30, and 40 gave similar results with more variation between the different strategies compared to between the number of clusters.

Results, Sample Sizes

As expected, a larger sample size led to increased classification accuracy, and a reduced number of misclassifications, see the cross-hatched yellow part in Fig. 4. But it was also clear that adding more training data sometimes increased the overall workload, in terms of grading and correcting. Sampling additional training instances did not always give to a corresponding decrease in the number of misclassifications. For example, for the cluster centroid strategy, the combined effort for grading training instances and correcting scores was at the lowest when only ten answers were graded as training data. For this setting, however, the number of instances requiring corrections was relatively larger compared to the size of the training data.

Results, Classification

Three different classifiers were evaluated, Random Forest, SVM, and a Complement Naive Bayes' (CNB) classifier. The classifier performance was evaluated using F-scores, accuracy, and quadratic kappa scores. The performance measures were calculated for each classifier, sampling size, and sampling strategy.

The accuracy score corresponds to the percentage of correctly classified answers, which is a meaningful score for this task, as the portion of scores requiring corrections corresponds to $1 - accuracy$. When instead focusing on comparing classifier performance for different strategies, it should be noted that the classes, the different scores assigned for the answers to each question, were highly imbalanced for some of the questions. In some cases, only a handful of answers got a partial score, while the majority of the answers were either given full or zero scores. Imbalanced data make the minority class, the class with very few examples, more difficult to classify correctly, and this is not fully reflected in the accuracy score. We therefore also report F-scores weighted by class size, and macro-weighted F-scores, see Fig. 5. A macro weighted average gives equal weight to each class and is, therefore, more influenced by the performance for minority classes compared to F-score weighted by class size.

In addition to these scores, quadratically weighted Cohen's kappa (QWK) was also calculated. This performance measure additionally takes the degree of incorrectness into account when the scores are calculated. A misclassification closer to the true score will be scored higher compared to a misclassification further from the true score.

Of the three classifiers, Random Forest (Breiman, 2001) performed slightly better overall. The average F-score using all strategies and sample sizes was 0.82 for Random Forest (0.79 for both CNB and SVM). The average macro-weighted

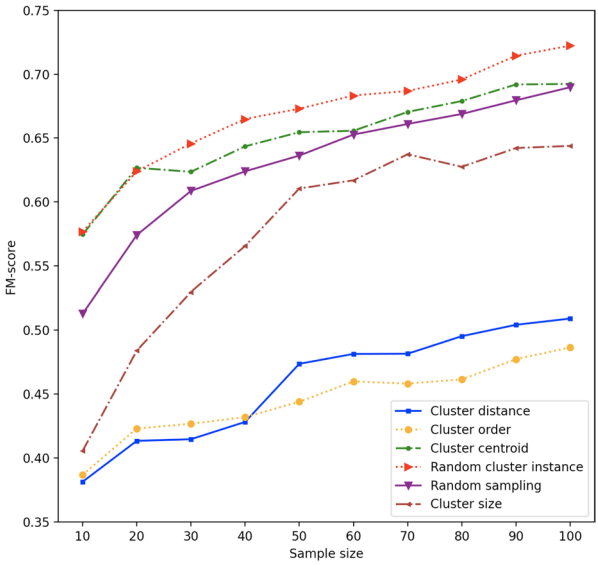


Fig. 5 Macro weighted F-scores for the different sampling strategies and sample sizes

F-score was 0.57 for Random Forest (0.54 for CNB and 0.56 for SVM), average QWK was 0.49 for CNB, 0.57 for SVM, and 0.56 for Random Forest. For accuracy, Random Forest had an average of 0.84, and the corresponding score for SVM and CNB was 0.81 and 0.80 respectively.

Table 3 shows the results for the different performance measures. The table also shows the share of answers that either was manually graded and used as training data (n_t), or incorrectly classified (n_c), and therefore in need of correction for each setting. For this measure, a lower score corresponds to an overall lower workload.

Three of the sampling strategies gave classifiers with an accuracy of $\geq 90\%$ but required different amounts of work to do so. For the two best cluster-based sampling strategies, 40 graded answers were required for this accuracy. For randomly sampling instances without first clustering the data, 80 answers were required to reach 90% accuracy.

The random cluster instance sampling strategy was found to be the overall best strategy, followed by sampling from cluster centroids. Next came the random sampling strategy, for which no clustering was performed. The remaining strategies, where full clusters were graded, resulted in less efficient classifiers.

The cluster centroid sampling strategy provides the best relative classifier performance for small sample sizes, likely because this strategy selects representative examples from each cluster. With larger sample sizes, the results are however better when sampling randomly over the different clusters, as this provides a more diverse sample compared to continuing to select instances close to the cluster centers. See Fig. 5 for an overview of classification results.

Table 3 Classification results. Size is the sample size. F-score is weighted by class size. MF shows the macro-weighted F-score. Cohen's quadratically weighted kappa is denoted by κ and $(n_t + n_c)/N$ shows the relative number of answers requiring manual work either through creating training data or correcting scores. Each result in this table was averaged over all 64 questions in the data set

Strategy	Size	F-score	Accuracy	MF-score	κ	$\frac{n_t+n_c}{N}$
Random cluster instance	10	0.84	0.85	0.58	0.59	0.18
Random cluster instance	20	0.87	0.88	0.62	0.65	0.19
Random cluster instance	30	0.88	0.89	0.65	0.68	0.22
Random cluster instance	40	0.89	0.90	0.66	0.69	0.25
Random cluster instance	50	0.90	0.90	0.67	0.69	0.28
Random cluster instance	60	0.90	0.91	0.68	0.70	0.31
Random cluster instance	70	0.90	0.91	0.69	0.69	0.35
Random cluster instance	80	0.91	0.92	0.70	0.69	0.38
Random cluster instance	90	0.91	0.92	0.71	0.70	0.42
Random cluster instance	100	0.91	0.92	0.72	0.69	0.45
Random sampling	10	0.80	0.82	0.51	0.49	0.21
Random sampling	20	0.85	0.86	0.57	0.59	0.21
Random sampling	30	0.87	0.88	0.61	0.63	0.23
Random sampling	40	0.88	0.89	0.62	0.66	0.26
Random sampling	50	0.88	0.89	0.64	0.68	0.29
Random sampling	60	0.89	0.90	0.65	0.69	0.32
Random sampling	70	0.89	0.90	0.66	0.70	0.36
Random sampling	80	0.90	0.90	0.67	0.71	0.39
Random sampling	90	0.90	0.91	0.68	0.71	0.42
Random sampling	100	0.90	0.91	0.69	0.73	0.46
Cluster order	10	0.54	0.61	0.32	0.35	0.41
Cluster order	20	0.59	0.64	0.39	0.39	0.41
Cluster order	30	0.67	0.71	0.44	0.49	0.38
Cluster order	40	0.72	0.75	0.47	0.57	0.38
Cluster order	50	0.74	0.77	0.50	0.60	0.40
Cluster order	60	0.77	0.79	0.53	0.59	0.41
Cluster order	70	0.78	0.80	0.56	0.60	0.44
Cluster order	80	0.78	0.80	0.56	0.58	0.48
Cluster order	90	0.78	0.81	0.57	0.58	0.51
Cluster order	100	0.79	0.81	0.60	0.61	0.54
Cluster centroid	10	0.85	0.86	0.58	0.71	0.17
Cluster centroid	20	0.87	0.88	0.62	0.73	0.19
Cluster centroid	30	0.87	0.88	0.61	0.72	0.23
Cluster centroid	40	0.88	0.90	0.65	0.74	0.26
Cluster centroid	50	0.89	0.90	0.65	0.73	0.30
Cluster centroid	60	0.89	0.90	0.65	0.74	0.33
Cluster centroid	70	0.89	0.90	0.66	0.74	0.37
Cluster centroid	80	0.89	0.90	0.66	0.74	0.41
Cluster centroid	90	0.89	0.91	0.68	0.74	0.45
Cluster centroid	100	0.89	0.91	0.69	0.73	0.49

Table 3 (continued)

Strategy	Size	F-score	Accuracy	MF-score	κ	$\frac{n_t+n_c}{N}$
Cluster size	10	0.64	0.67	0.41	0.32	0.35
Cluster size	20	0.73	0.76	0.48	0.44	0.30
Cluster size	30	0.78	0.80	0.53	0.52	0.30
Cluster size	40	0.81	0.82	0.57	0.56	0.31
Cluster size	50	0.82	0.84	0.61	0.59	0.33
Cluster size	60	0.84	0.86	0.62	0.60	0.35
Cluster size	70	0.86	0.87	0.64	0.62	0.38
Cluster size	80	0.86	0.87	0.63	0.61	0.41
Cluster size	90	0.87	0.88	0.64	0.60	0.44
Cluster size	100	0.87	0.88	0.64	0.58	0.48
Cluster distance	10	0.64	0.69	0.38	0.27	0.34
Cluster distance	20	0.68	0.72	0.41	0.33	0.34
Cluster distance	30	0.68	0.72	0.41	0.32	0.37
Cluster distance	40	0.71	0.74	0.43	0.36	0.38
Cluster distance	50	0.75	0.77	0.47	0.43	0.39
Cluster distance	60	0.76	0.78	0.48	0.45	0.41
Cluster distance	70	0.75	0.77	0.48	0.46	0.45
Cluster distance	80	0.77	0.79	0.50	0.48	0.47
Cluster distance	90	0.78	0.80	0.50	0.50	0.49
Cluster distance	100	0.79	0.80	0.51	0.51	0.52

Overall Reduction of Workload

To measure the overall reduction in workload, the time spent on grading an exam with and without support was compared. Grading a complete with support can be described by $\lambda_t n_t + \lambda_a n_a + \lambda_c n_c$.

The timed grading was used to estimate the time required for the different grading actions. During the timed grading, the cluster centroid strategy was used together with a sample size of 20, and this strategy gave an accuracy of 0.88. In the timed experiments, group D was most similar to the overall grading approach. This group was therefore used to estimate the value of λ_t, the time for grading training data (8.8 seconds on average), and λ_a, the time for reviewing correct scores (0.35 seconds on average).

In group C, the review and correction of scores were timed separately, which made it possible to estimate λ_c, as the sum of the time it took to identify an incorrect score (2.1 seconds on average) and correct such scores (8.8 seconds on average).

Using these estimates, grading a complete exam without support would take 36.0 hours, and grading with support would take 9.2 hours, saving 74% of the time spent on grading. Note that λ_t, during the first grading phase, was estimated to be the same for grading with and without support since only one instance was sampled from each cluster, which means that the answers were not displayed in clusters during the first grading phase of the timed grading.

Reviewing correctly scored answers was found to be very fast for questions where the classifier had suggested correct scores for all answers. It might, however, be the case that a teacher and a trained classifier find the same type of answers "easy" to grade, whereas more complex answers could take more time. The 2.4 seconds required, on average, for both reviewing and correcting the classified scores for group D could, therefore, be considered as an upper bound on λ_a. This gives 2.4 seconds as an estimate of both λ_c and λ_a. Using these values instead, the overall time saved would instead be 64%.

Other Results

Clustering served two purposes in this work, to simplify the grading, and to guide the sampling strategies. As previous works show, grouping student answers based on similarity can make grading more efficient as groups of answers can be graded together rather than grading each answer individually. Presenting similar answers together to a grader might potentially also support fair grading. In this case, the exams were graded by two graders who both reported that using clustering made the grading more efficient in terms of time spent on each answer and that the clusters facilitated a consistent scoring of answers since all answers to one question were displayed simultaneously to the grader. Using a spreadsheet program for all manual work was found to be practical and eliminated any need to develop a custom graphical user interface. It allowed for an easy overview of all answers and the graders could assign both scores and comments to the answers.

Question demoting is a method for removing information given in a question from a student answer (Mohler et al., 2011). The idea behind question demoting is that student answers should be graded on whether they contain correct information relevant to the question. Information given in the question and repeated in the answer should not influence the score given to that answer. Question demotion was introduced by Mohler et al. (2011), and Basu et al. (2013) also employed question demoting by treating the words in the questions as stop words. We evaluated stop-word-based question demoting for our data set, but did not observe any improvement in the classification results.

Answers to one exam question could be considered incorrect answers to all other questions. It was therefore evaluated whether it was possible to supplement the training data for one question with answers to other questions by labeling those answers as incorrect. This approach did improve classification results for a few questions but did not improve classification results overall.

The classification performance correlated with the student performance. The classifier was more accurate with relatively more high-quality answers to learn from and less accurate for questions where the students struggled. This was perhaps partly due to there being more variation among incorrect answers compared to correct answers, making it more difficult to learn how to separate correct and incorrect answers.

Horbach and Zesch (2019) measured the impact of different types of variances in student answers for automatic grading. They found that shorter answers are generally easier to score automatically. Zesch et al. (2015) also noted that clustering

was more efficient for reducing effort in the grading of short answers. In our work, questions with shorter answers were easier to score correctly in some cases, but no statistically significant correlation was found between average answer length for a question and the classifier performance on the same question.

Concluding Remarks

Conclusions

The main goal of this work was to reduce the workload in exam grading for exams consisting of questions which can be answered with relatively short, free-text answers. A grading approach in two steps—using both clustering and classification—was evaluated along with several sampling strategies for selecting training data.

Using an appropriate sampling strategy was found to have a large impact on the amount of manual work required to grade the complete exam. Cluster-based sampling, where a subset of instances of each cluster was selected as training data, gave the largest reduction in the workload of the evaluated strategies, but even the weakest strategy substantially reduced the overall time spent on grading. If only one answer from each cluster is graded, selecting the answer closest to the center of the cluster is reasonable, but the best performance was consistently achieved with the strategy where random instances were sampled from the different clusters (random cluster instance). For the sample size, grading on average 30 answers for each question provided a balance between the initial grading and score corrections for the data set in this study.

Timing the grading process confirmed that displaying groups of similar answers to the grader saved time and effort during grading. Clustering similar answers together on average reduced the time spent on grading by 69%. Similarly, it was found that providing score suggestions to the grader also reduced the time spent on scoring individual answers by a comparable amount. This of course requires that the score suggestions are largely correct; we found a considerable difference between the effort required in correcting an incorrect score (λ_c) compared to reviewing correct scores (λ_a). This confirms the initial assumption that the workload for reviewing a correct answer is small compared to grading an answer.

Combining both clustering and score suggestions provided a further reduction in time spent on reviewing, reducing the time spent on grading by 73%. It can also be noted that there were slightly more corrections required for the group of questions with both clustering and automatic scoring applied, compared to the group with score suggestions only, which possibly also affected the timing for this group.

It can therefore be concluded that both the clustering and the classification of answers reduced the effort required for short answer grading and that it is possible to reduce the effort associated with grading while still reviewing all automatically assigned scores. This manual review was essential as any automated assessment which affects a student's grade could be considered as "high-stakes".

Discussion

The overall results of this work showed that this approach can be used to reduce workload in exam grading. But how to best divide the work between grading answers, to be used as training data, and correcting the output of the classifier can, however, be a matter of taste. While correcting an incorrect score took about as much time as adding an extra instance to the training set, it could be preferable for a teacher to spend relatively more time on grading rather than correcting scores.

Similarly, sampling strategies where full clusters were graded in the first grading phase resulted in a lower classifier performance compared to strategies where training instances were sampled from different clusters. While grading answers grouped by a clustering algorithm saved time and effort, it is likely preferable to work on full clusters only during the review phase. The first grading phase should instead aim to cover a set of diverse answers, to reduce the sample size without compromising the overall results.

Our approach also produces data consisting of fully graded exam questions. If a subset of questions is reused in subsequent course iterations, more work can be saved, since no new manual grading of training data will be required for those questions.

The time spent on grading individual answers can also depend on the type of question. When grouping the answers according to Bloom's taxonomy, it was found that answers to questions testing the student's ability to apply procedural knowledge were quick to grade, likely because these answers corresponded to the result of a calculation with a clear distinction between correct and incorrect answers. Questions requiring a deeper understanding also allow for more variation in the answers, where, in particular, partially correct answers can require extra consideration.

An alternative to using the cluster-based sampling evaluated here could be active learning. Active learning can also be combined with cluster-based sampling, such as was done by Kang et al. (2004), who used cluster-based sampling to create the initial training set for an active learning algorithm. Active learning, however, requires retraining of the machine learning model in each iteration, which can be impractical, specifically when the grading is performed by more than one person in parallel.

Future Work

While the overall setup for exam grading evaluated here was found to decrease the time spent on exam grading, there are possible improvements. A limitation of this work is that inter-annotator agreement has not been calculated. We intend to include this in future work.

The possible improvements also include added functionality for the manual grading. For example by allowing the grader to search for key terms among the answers, automatic handling of identical answers, and dedicated methods for answers containing calculations. Focusing the effort on questions and answers where the classifier is less confident is also left for future work.

Since the different parts of the overall approach, the sampling strategies, the manual grading, the classification, and the review of the classification are practically independent of each other, each part can be improved separately to increase the overall efficiency. The timing of the grading however clearly showed that accurate classification should be prioritized, as this provides the largest potential for further reduction of workload.

Author Contributions Not applicable.

Funding Open access funding provided by Stockholm University. No external funding (funded by the Department of Computer and Systems Sciences at Stockholm University).

Availability of data and material The datasets generated during and analysed during the current study are available in the Harvard Dataverse repository, https://doi.org/10.7910/DVN/NMEM7D

Code availability Not applicable.

Declarations

Conflicts of interest The authors declare that they have no conflict of interest.

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

References

Anderson, L. W., Bloom, B. S., & et al. (2001). *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*. Longman.

Azad, S., Chen, B., Fowler, M., West, M., & Zilles, C. (2020). Strategies for deploying unreliable ai graders in high-transparency high-stakes exams. In: International Conference on Artificial Intelligence in Education. Springer, pp 16–28

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.

Bonthu, S. (2021). Automated short answer grading using deep learning: A survey. In: Machine Learning and Knowledge Extraction: 5th IFIP TC 5, TC 12, WG 8.4, WG 8.9, WG 12.9 International Cross-Domain Conference, CD-MAKE 2021, Virtual Event, August 17-20, 2021, Proceedings, Springer Nature, vol 12844, p 61

Breiman, L. (2001). Random forests. *Machine learning, 45*(1), 5–32.
Brooks, M., Basu, S., Jacobs, C., & Vanderwende, L. (2014). Divide and correct: using clusters to grade short answers at scale. In: Proceedings of the first ACM conference on Learning@ scale conference, pp 89–98
Burrows, S., Gurevych, I., & Stein, B. (2015). The eras and trends of automatic short answer grading. *International Journal of Artificial Intelligence in Education, 25*(1), 60–117.
Cohen, J. (1968). Weighted kappa: nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin, 70*(4), 213.
Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805
Filighera, A., Steuer, T., Rensing, C. (2020). Fooling automatic short answer grading systems. In: International Conference on Artificial Intelligence in Education. Springer, pp 177–190
Galhardi, L., de Souza, R. C. T., & Brancher, J. (2020). Automatic grading of portuguese short answers using a machine learning approach. In: Anais Estendidos do XVI Simpósio Brasileiro de Sistemas de Informação, SBC, pp 109–124
Geigle, C., Zhai, C., & Ferguson, D. C. (2016). An exploration of automated grading of complex assignments. In: Proceedings of the Third (2016) ACM Conference on Learning@ Scale, pp 351–360
Gomaa, W. H., & Fahmy, A. A. (2014) Arabic short answer scoring with effective feedback for students. International Journal of Computer Applications 86(2)
Horbach, A., & Palmer, A. (2016) Investigating active learning for short-answer scoring. In: Proceedings of the 11th workshop on innovative use of NLP for building educational applications, pp 301–311
Horbach, A., & Pinkal, M. (2018). Semi-supervised clustering for short answer scoring. In: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), pp 4066–4071
Horbach, A., & Zesch, T. (2019). The influence of variance in learner answers on automatic content scoring. In: Frontiers in Education, Frontiers, vol 4, p 28
Horbach, A., Palmer, A., & Wolska, M. (2014). Finding a tradeoff between accuracy and rater's workload in grading clustered short answers. In: LREC, Citeseer, pp 588–595
Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering, 9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55
Kang, J., Ryu, K. R., Kwon, H. C. (2004) Using cluster-based sampling to select initial training set for active learning in text classification. In: Pacific-Asia conference on knowledge discovery and data mining, Springer, pp 384–388
Ke, Z., & Ng, V. (2019). Automated essay scoring: A survey of the state of the art. *IJCAI, 19*, 6300–6308.
Kishaan, J., Muthuraja, M., Nair, D., & Plöger, P. G. (2020). Using active learning for assisted short answer grading. In: ICML 2020 Workshop on Real World Experiment Design and Active Learning
Kulkarni, C. E., Socher, R., & Bernstein, M. S., Klemmer, S. R. (2014). Scaling short-answer grading by combining peer assessment with algorithmic scoring. In: Proceedings of the first ACM conference on Learning@ scale conference, pp 99–108
Kumar, S., Chakrabarti, S., & Roy, S. (2017). Earth mover's distance pooling over siamese lstms for automatic short answer grading. In: IJCAI, pp 2046–2052
Leacock, C., & Chodorow, M. (2003). C-rater: Automated scoring of short-answer questions. *Computers and the Humanities, 37*(4), 389–405.
Lun, J., Zhu, J., Tang, Y., & Yang, M. (2020). Multiple data augmentation strategies for improving performance on automatic short answer scoring. *Proceedings of the AAAI Conference on Artificial Intelligence, 34*, 13389–13396.
Madnani, N., & Cahill, A. (2018). Automated scoring: Beyond natural language processing. In: Proceedings of the 27th International Conference on Computational Linguistics, pp 1099–1109
Marvaniya, S., Saha, S., Dhamecha, T.I., Foltz, P., Sindhgatta, R., & Sengupta, B. (2018). Creating scoring rubric from representative student answers for improved short answer grading. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, pp 993–1002
Mieskes, M., & Pado, U. (2018). Work smart-reducing effort in short-answer grading. In: Proceedings of the 7th Workshop on NLP for Computer Assisted Language Learning (NLP4CALL 2018) at SLTC, Stockholm, 7th November 2018, Linköping University Electronic Press, 152, pp 57–68
Mohler, M., Bunescu, R., & Mihalcea, R. (2011). Learning to grade short answer questions using semantic similarity measures and dependency graph alignments. In: Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies, pp 752–762

Nielsen, R. D., Ward, W. H., & Martin, J. H. (2008). Learning to assess low-level conceptual understanding. In: Flairs conference, pp 427–432

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.

Qi, P., Zhang, Y., Zhang, Y., Bolton, J., & Manning, C. D. (2020). Stanza: A Python natural language processing toolkit for many human languages. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations. https://nlp.stanford.edu/pubs/qi2020stanza.pdf. Accessed Sep 2020.

Reimers, N., Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, http://arxiv.org/abs/1908.10084

Reimers, N., & Gurevych, I. (2020). Making monolingual sentence embeddings multilingual using knowledge distillation. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, https://arxiv.org/abs/2004.09813

Riordan, B., Horbach, A., Cahill, A., Zesch, T., & Lee, C. (2017). Investigating neural architectures for short answer scoring. In: Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications, pp 159–168

Settles, B. (2009). Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison

Singhal, A., et al. (2001). Modern information retrieval: A brief overview. *IEEE Data Eng Bull, 24*(4), 35–43.

Souza, D. M., Felizardo, K. R., & Barbosa, E. F. (2016). A systematic literature review of assessment tools for programming assignments. In: 2016 IEEE 29Th international conference on software engineering education and training (CSEET), IEEE, pp 147–156

Sung, C., Dhamecha, T. I., & Mukhi, N. (2019) Improving short answer grading using transformer-based pre-training. In: International Conference on Artificial Intelligence in Education, Springer, pp 469–481

Yen, S. J., & Lee, Y. S. (2009). Cluster-based under-sampling approaches for imbalanced data distributions. *Expert Systems with Applications, 36*(3), 5718–5727.

Zesch, T., Heilman, M., & Cahill, A. (2015). Reducing annotation efforts in supervised short answer scoring. In: Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications, pp 124–132