International Journal of Artificial Intelligence in Education manuscript No.
(will be inserted by the editor)

An AI-based system for formative and summative assessment in data science courses

Pierpaolo Vittorini, Stefano Menini, Sara Tonelli

the date of receipt and acceptance should be inserted later

Vittorini Pierpaolo (corresponding author)
University of L'Aquila
P.le S. Tommasi 1
67100 Coppito, L'Aquila, Italy
Tel.: +39 0862 434650
E-mail: pierpaolo.vittorini@univaq.it
ORCID: 0000-0002-6975-8958

Menini Stefano
Fondazione Bruno Kessler
via Sommarive 18
Trento, Italy
E-mail: menini@fbk.eu
ORCID: 0000-0002-4296-4743

Tonelli Sara
Fondazione Bruno Kessler
via Sommarive 18
Trento, Italy
E-mail: satonelli@fbk.eu
ORCID: 0000-0001-8010-6689

Abstract Massive open online courses (MOOCs) provide hundreds of students with teaching materials, assessment tools, and collaborative instruments. The assessment activity, in particular, is demanding in terms of both time and effort; thus, the use of artificial intelligence can be useful to address and reduce the time and effort required. This paper reports on a system and related experiments finalised to improve both the performance and quality of formative and summative assessments in specific data science courses. The system is developed to automatically grade assignments composed of R commands commented with short sentences written in natural language. In our opinion, the use of the system can (i) shorten the correction times and reduce the possibility of errors and (ii) support the students while solving the exercises assigned during the course through automated feedback. To investigate these aims, an ad-hoc experiment was conducted in three courses containing the specific topic of statistical analysis of health data. Our evaluation demonstrated that automated grading has an acceptable correlation with human grading. Furthermore, the students who used the tool did not report usability issues, and those that used it for more than half of the exercises obtained (on average) higher grades in the exam. Finally, the use of the system reduced the correction time and assisted the professor in identifying correction errors.

Keywords Assessment · Automated grading · NLP · ML · Embeddings · SVM

Declarations

A) Funding (information that explains whether and by whom the research was supported)

This research was funded by internal grants from the University of L'Aquila and Fondazione Bruno Kessler.

B) Conflicts of interest/Competing interests (include appropriate disclosures)

There are no conflicts of interest or competing interests to declare.

C) Availability of data and material (data transparency)

All data and material used in the paper are available at:
https://vittorini.univaq.it/uts/

D) Code availability (software application or custom code)

The source code has not yet been released.

1 Introduction

Massive open online courses (MOOCs) are designed to provide students with teaching materials, assessment tools, and collaborative instruments. For the assessment activity, the formative assessment occurs during the execution of a course to verify the students' learning progress (by the teacher or by the students themselves), whereas summative assessment occurs at the end to determine the learning outcomes [28]. The manual grading of assignments is a tedious and error-prone task, and the problem particularly aggravates when such an assessment involves a large number of students, such as in MOOCs. In such a context, the use of artificial intelligence can be useful to address these issues [36]. This will automate the grading process to assist teachers in the correction and enable students to receive immediate feedback, thus improving their solutions before the final submission.

Considering this, our specific problem regards the automated grading of assignments, the solutions of which are composed of a list of commands, their outputs, and possible comments. The proposed approach addresses the problem by defining the distance between the solution given by the student and the correct solution (given by a professor).

The study focuses on assignments, the solutions of which constitute commands written in R language [45], their outputs, and explanations of the results through short answers written in the Italian language. The implementation relies on state-of-the-art natural language processing combined with a code analysis module to build an interactive assessment tool. In our opinion, the use of the tool can (i) shorten the correction times and reduce the possibility of errors, (ii) support the students while solving the exercises assigned during the course through automated feedback, and (iii) increase the learning outcomes of the students that use the system as a formative assessment tool.

To investigate these aims, an ad-hoc experiment was conducted for three courses containing the specific topic of statistical analysis of health data, during which the tool was used first for formative assessment and then for summative assessment. In particular, students were invited to use the tool for formative assessment while doing their homework, use the automated feedback, and report on the system usability. At the end of the homework, a professor provided the manual (and final) feedback. As a summative assessment tool, the exams were then randomly corrected either manually or through the automated system. We measured the performance of the binary classifier for predicting the correctness of the short sentences, overall quality of the automated system to predict the manual grading (i.e., code and short sentences), system usability, correction performance, and the impact of the use of the system in terms of the students' final grading.

The remainder of this paper is structured as follows. Section 2 discusses the application scenario and educational influence of the proposal, i.e., the specific problem that suggested the general approach, and related implementation. Section 3 summarises the related work on the automated grading of open-ended sentences and code-snippets. Section 4 describes the classification and grading processes; it details the data, experimental setup, features used to classify the student answers as correct or incorrect, related results in terms of accuracy, F1, Cohen's K, and the system's ability to predict the manual grade. Sections 5 and 6 present the results regarding the use of the tool during formative and summative assessment activities, respectively. Section 7 discusses the results, highlighting the identified limitations. Finally, Section 8 summarises the work and key results.

2 Application scenario and educational impact

The Health Informatics course in the Master's degree program in Medicine and Surgery at the University of L'Aquila (Italy) is organised in two parts. The first part regards the theoretical aspects of health informatics, while the second part is practical and focuses on the execution of statistical analyses in R and correct interpretation of the results in the corresponding clinical findings. The practical part is the only topic for the Information Systems course for the two Master degrees in Prevention Sciences and Nursing Sciences.

With specific regard to the practical part of the three courses, students complete several exercises as homework, after having attended the lessons, and finish the part with a practical assessment. The exercises and final exam have the same structure: they begin with the definition of the dataset and list the analyses and technical/clinical interpretations that should be performed. The analyses must be performed through R commands and can be descriptive (e.g., mean, sd), inferential (e.g., t.test, wilcox.test), and for testing normality (e.g., shapiro.test). For the interpretation of the results, students must be able to understand, e.g., whether the test for normality suggests that the distribution should be considered normal, or whether a hypothesis testing is statistically significant.

For example, let us consider the following assignment:

Exercise 1 *Consider the following dataset:*

Subject	*Surgery*	*Visibility*	*Days*
1	*A*	*7*	*7*
	...		
10	*B*	*16*	*12*
	...		
20	*C*	*19*	*4*

The data included 20 subjects (variable Subject) that underwent three different surgical operations (variable Surgery). We observed the scar visibility (variable Visibility) in terms of ranks ranging from "1" (best) to "20" (worst). We also measured the hospital stay (variable Days).

1. *Calculate the mean (with confidence intervals) and standard deviation of the hospital stay.*
2. *Calculate the absolute and relative frequencies (with confidence intervals) of the surgical operations.*
3. *Verify whether the hospital stay can be considered as extracted from a normal distribution.*
4. *Comment on the result.*
5. *Calculate the median, 25^{th}, and 75^{th} percentile of the hospital stay for the different surgical operations.*
6. *Verify whether the aspect of the scar is different among the different surgical operations.*
7. *Comment on the result.*

Submit as a solution, text containing the list of R commands with the respective output and your interpretation of the analyses of Items 3 and 6.

Concerning Item 6, because the scar visibility is qualitative and the number of different surgeries is three, the student should use a Kruskal–Wallis test. Such a test is executed in R through the command

```
kruskal.test(data = exam, Visibility ~ Surgery)

    Kruskal—Wallis rank sum test

data:  Visibility by Surgery
Kruskal—Wallis chi—squared = 9.8959, df = 2, p—value = 0.007098
```

Listing 1 Solution to Item 6 of Exercise 1

From the p-value, which is less than 0.05, the student can then conclude that it is highly unlikely that the difference in Visibility occurred by chance, and therefore, it could be a consequence of the different surgeries. This conclusion is the solution to Item 7 of the assignment.

Even though the results reported in this paper focus on assignments regarding the statistical analysis of health data in R, this proposal can be applied to other domains and languages such as the following problem of classification in a hypothetical course in machine learning.

Exercise 2 *We have a classification problem to be addressed using a support vector machine (SVM) classifier. Let us suppose that we have the model saved in file* `svm_model.mat` *and the features in file* `features.csv`*. In MATLAB, we load the necessary files, apply the classification, and interpret the result.*

The solution is straightforward:

```
>> load('svm_model.mat')
>> x = csvread('features.csv')

x =

     0     0

>> predict(svm_model,x)

ans =

     1

>> % The classifier considered the sentence to be incorrect
```

Listing 2 Example of assignment regarding machine-learning solved in MATLAB

Given such an educational scenario, the proposed tool supports both formative and summative assessment as follows (see Figure 1). As a formative assessment instrument, it can provide students with both textual feedback and an estimated evaluation of the quality of the submitted solution, and it enables teachers to monitor the students' progress with the homework. At this point, the feedback provided to the students regards which commands were correctly issued, which commands contained an error either in the call or in the passed data, which commands were missed, and whether the comments were correct. The feedback can be requested by the student during the execution of the homework. As a summative assessment instrument, the tool can be used by the teacher to shorten and improve the manual correction activities. Accordingly, this has several expected educational benefits. For students, these include the ability to support their understanding of the commands and interpretation of the results. Consequently, students should be able to improve their

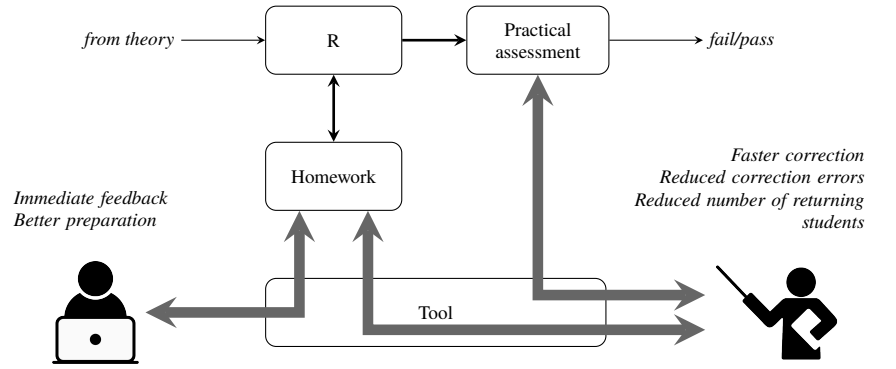

Fig. 1 Application scenario and educational impact

final learning outcomes. Accordingly, the professors should expect a decreased number of returning students. Specifically for professors, the tool should be able to reduce their workload in terms of both evaluation time and errors.

3 Background

Several solutions have been previously proposed to perform automated grading of open-ended and code-snippet answers [12,48].

For those addressing open-ended answers, the common task is to assign either a real-valued score (e.g., from "0" to "1") or to assign a label (e.g., "correct" or "irrelevant") to a student response. However, these attempts have mainly focused on English, whereas our courses and exams are in Italian. Several existing approaches to short-answer grading rely on knowledge bases and syntactic analyses [44], which are available only for a limited set of languages. More recent studies have exploited the potential of vector-based similarity metrics to compare the students' answers against a gold standard [49], similar to the distance-based features described in Section 4.2. Such features have been further explored in [41] to include a number of corpus-based, knowledge-based, word-based, and vector-based similarity indices. Recently, few studies have been presented using transformer-based architectures [20] and neural network classifiers to perform short-answer grading [39,51]. Although neural approaches have demonstrated acceptable performance and generalisation capabilities across domains [50] and languages [13], they typically require large amounts of data to learn an accurate classification model, as confirmed in the studies cited above. Because our dataset consists of only approximately 1,000 training examples, in our case we adopt an embedding-based approach to represent them using a non-neural classifier.

For code-snippets questions, the automated assessment of student programming assignments was first attempted in the sixties [29]. Currently, the available tools use either manual, automated, or semi-automated approaches. The manual assessment tools assist the teacher in assessing the students' assignments; however, the assessment itself is performed manually by the professor [17]. Automated assessment tools proceed autonomously through the evaluation [14,8]. Semi-automated tools require the instructor to perform additional manual inspections to validate doubtful cases [31]. Furthermore, the available tools can either follow an instructor-centred, student-centred, or hybrid approach, based on whether they support the teacher, student, or both, respectively. Moreover, the available tools can offer a prelimi-

nary validation of the solution [40], a partial feedback, e.g., by displaying the results of the automated grading before the final submission [21], or can include the instructor review, i.e., the partial feedback specifies that the solution will be reviewed by the teacher [25]. In terms of the correction strategy, the available tools either compile and execute the submitted code against test data (to compare the obtained results with the expected ones) [34] or analyse the students' source code statically to understand its correctness [22]. In terms of programming languages, most of these solutions focus on Java, C, or C++ [48]. In the specific context of MOOCs, several systems have been reported [23,19,5]. Unfortunately, to the best of our knowledge, none of the systems focus on R, which is the programming language in which we are interested.

In this context, in [3], we address the problem discussed in Section 2 by introducing an approach valid for assignments, the solutions of which can be represented as a list of triples containing the command, its output, and a possible comment. In the proposed approach, a solution provided by a student is compared with the correct solution given by the professor. Hence, a student can

- provide a correct command that returns the correct output;
- provide a command with an error either in the call (e.g., a t.test command without the required paired=TRUE option) or in the passed data. In both cases, the returned output is different from that of the professor.
- miss the command;
- interpret the result in the correct/incorrect manner.

Thus, we count the number of missing commands (i.e., commands that are in the correct solution, however not in the submitted solution), number of commands with a different output (i.e., commands that are in both solutions, yet differ in terms of the output), and whether the $i-th$ comment can be considered correct. We then define the distance between the two solutions as follows:

$$d = w_m \cdot M + w_d \cdot D + \sum_i w_c \cdot (1 - C_i), \tag{1}$$

where w_m is the weight assigned to the missing commands; M is the number of missing commands; w_d is the weight assigned to the commands with different outputs; D is the number of commands with different outputs; w_c is the weight assigned to the distance between the comments, and $C_i = 1$ if the comment is correct or $C_i = 0$, otherwise. Clearly, if the number of missing commands, commands with different outputs, and incorrect comments is high, the distance is large. In the final step, the distance is converted into the final grade. In Italy, a grade is a number ranging from "0" to "30", plus a possible additional mention of "cum laude" customarily considered as "31". An exam is passed with a grade greater than or equal to "18". Accordingly, with a simple proportion, the tool converts a distance equal to zero to the grade of "31", and a maximum distance (that depends on the number of commands and comments required to solve the assignment) as the grade of zero.

For example, we consider the sample assignment discussed in Section 2 and assume the weights $w_m = 0.051$, $w_d = 0.014$, and $w_c = 0.03$ (Subsection 4.4.2 discusses how these were experimentally calculated). The solution requires 15 commands and 2 interpretations of the normality and hypothesis tests. The best solution is the one with $M = 0$, $D = 0$, and $C_1 = C_2 = 1$, which results in a distance $d_b = 0$ and grade $G_b = 31$. Conversely, the worst solution is the empty solution, with $M = 15$, $D = 0$, and $C_1 = C_2 = 0$, with a distance of $d_w = 0.825$ and grade $G_w = 0$. Therefore, the grade G is calculated from distance d as follows:

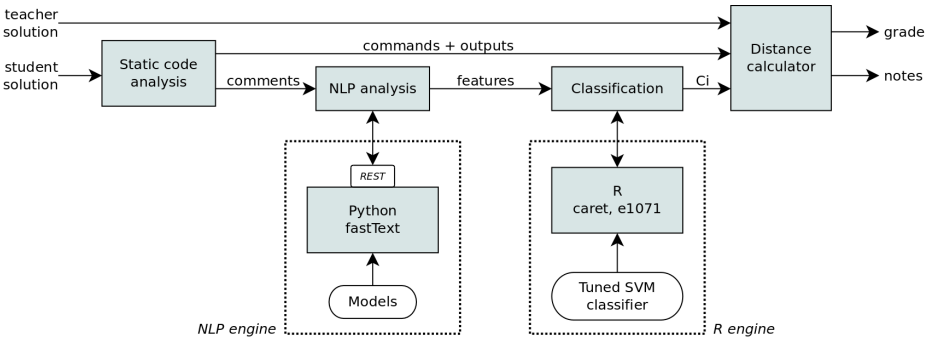

Fig. 2 Tool architecture

$$G = 31 \cdot \left(1 - \frac{d}{0.825}\right).$$

Accordingly, a sufficient solution, i.e., with a grade of "18", requires a distance $d_s = 0.345$, that can be obtained, for example, with $M = 6$, $D = 3$, and $C_1 = C_2 = 0$. That is, a student can pass the exam by submitting six (over 15) correct commands and three commands with a mistake either in the command or passed data, without providing an interpretation of the tests.

The aforementioned approach is implemented in a tool with the following characteristics: it provides an automated grading of assignments, follows a hybrid approach, uses static source code analysis for the code snippets and a supervised classifier based on sentence embeddings for the open-ended answers, provides partial feedback to the students, and includes an instructor review. The tool focuses on assignments such as those discussed in Section 2, i.e., with solutions implemented as a set of R commands and comments written in the Italian language. In a preliminary implementation [3], the weights were fixed to $w_m = 1$, $w_d = 0.1$, and $w_c = 0.1$, and C_i was calculated as the Levenshtein string similarity distance between the student answers and correct answers [37], divided by the length of the longest string, to return a distance in the range [0,1]. It is clear that string similarity captures the lexical rather than semantic similarity [26]; therefore, it can assign a low similarity score to strings conveying the same concept with a different lexical choice (and vice versa). For example, given the sentence "The difference is statistically significant", the adopted string similarity distance would consider the sentence "The difference is not statistically significant" (that conveys the opposite concept) only 0.085 distant. However, the tool demonstrated an acceptable ability to predict the grade given by the professor ($R^2 = 0.740$) [3]. In [18], we reported on an experiment that demonstrated the potential of a supervised classification approach for capturing semantic similarity between two strings using sentence embeddings, which resulted in acceptable F1 and accuracy values.

The experiment is fully detailed in this paper, i.e., the optimal weights are derived with a maximum-likelihood estimation procedure. The approach of generating the embeddings and using a supervised classifier was compared with BERT [20] and a baseline consisting of only distance-based features. The tool was implemented in the UnivAQ Test Suite [7] (i.e., an assessment system developed to support a wide range of questions, with adaptive capabilities [2] and high flexibility, implementing artificial intelligence methods, with a responsive interface) and has been extensively tested with students. The tool implementation is displayed in Figure 2. As indicated, the tool grades the assignments as follows: (i) executes

the static analysis of the student solution (i.e., by extracting the list of commands, outputs, and possible comments (static code analysis module)), (ii) processes the comments through the natural language processing engine (NLP analysis module), (iii) classifies the comments as correct/incorrect through the R engine (classification module), and (iv) measures the distance and assembles the notes (distance calculator module). The NLP engine converts each comment into a sentence embedding aimed at providing a semantic representation of the content. This is also undertaken for the correct answer, against which the student's comment is compared. Moreover, another set of similarity-based features is computed by measuring the distance between the comment and the correct answer (for details, see Section 4.2). The NLP engine is implemented as a Python server and called by the tool via a REST interface. The reason behind this implementation is that the language models used by the engine to generate the embeddings are reasonably large, and loading them into the engine would require time. Therefore, with such a solution, the models are loaded only when the server starts and then used when the embeddings are generated. Finally, the R engine receives the features from the NLP engine and uses CARET and E1071 [33,43] to perform the classification.

4 Classification and grading

As mentioned in the previous section, the tool must analyse the code and classify the comments. For the static code analysis, we refer to Listing 3 (part of the solution given to the assignment presented in Section 2).

```
1 > kruskal.test(data = exam, Visibility ~ Surgery)
2
3     Kruskal—Wallis rank sum test
4
5 data:  Visibility by Surgery
6 Kruskal—Wallis chi—squared = 9.8959, df = 2, p—value = 0.007098
7
8 # The difference in visibility between the different surgeries was statistically
      ↪ significant because p < 0.05
```

Listing 3 Sample command, output and comment

Extracting the command (line 1), respective output (lines 3–6), and comment (line 8) is straightforward, as is the comparison of the command and output with the solution given by the professor (e.g., whether it is the same command or whether it has produced the same output).

The comment is passed to the NLP engine, which extracts the features that are then used by the R engine for classification. For this purpose, a supervised model was trained to build a classifier that could determine whether a comment was correct or did not give the correct answer. In general, the goal of this classification step was to capture the similarity between the student's answer and correct answer, and a high similarity led to a pass judgement. Two real examples from the dataset are reported below, translated into English.

Correct answer:
(Student) *the p-value is less than 0.05; hence, the result on the sample is statistically significant and can be extended to the population.*

Incorrect answer:
(Student) *the p-value is > 0.05; hence, the result is statistically significant, or in other words, the wound appearance does not depend on the type of surgery.*

(Gold) *Given that the p-value is < 0.05, I can generalise the result observed in my sample to the population; hence, there is a statistically significant difference in the appearance of the wound between Surgery A and Surgery B.*

The task was extremely challenging as the sentence pairs in the dataset demonstrated a high level of word overlap, and the discriminant between a correct and incorrect answer could be only the presence of "<" instead of ">", or a negation.

4.1 Data

To train the classification model, we built the dataset available at [1], which contains a list of comments written by students with a unique ID, their type (if given for the hypothesis or normality test), their 'correctness' in a range from "0" to "1", their fail/pass result, accompanied by (i) the gold standard (i.e., the correct answer) and (ii) an alternative gold standard. All students' answers were collected from the results of real exams; all gold standard answers, the correctness score, and pass/fail judgment were assigned by the professor who evaluated such exams.

To increase the number of training instances and achieve a superior balance between the two classes, we also manually negated a set of correct answers and reversed the corresponding fail/pass result, thereby adding a set of (iii) negated gold standard sentences. The sentences of all students and gold standard sentences were in the Italian language. In summary, the dataset contained 1,069 student/gold standard answer pairs, 663 of which were labelled as "pass" and 406 as "fail".

4.2 Feature description

The NLP engine returns two different types of features to assess the similarity between two sentences. The first set of features consists of the *sentence embeddings* of the two solutions to represent the semantics of the two texts in a distributional space. The second consists of *distance-based features* between the student's comment and the correct answer to assess the similarity of the two sentences at the lexical and semantic levels.

For the creation of the embeddings, we relied on fastText [9,27], a library developed at Facebook to learn word representations and convert them easily into sentence representations if text snippets are given in the input, as in our case. One of the main advantages of fastText is the fact that several pre-trained models for different languages are made available from the project website, without the requirement to retrain such computationally intensive models. More importantly, it can address rare words using subword information, such that the issue of unseen words is mitigated. The representation of the sentences is then obtained by combining vectors encoding information on both words and subwords. For our task, we adopted the precomputed Italian language model[1] trained on Common Crawl and Wikipedia. This was particularly suitable for our task as Wikipedia includes scientific and statistics pages, making the specific domain of the exam under consideration well represented by the model. The

[1] https://fasttext.cc/docs/en/crawl-vectors.html.

embeddings were created using continuous bag-of-words with position-weights, a dimension of 300, character n-grams of length 5, and a window of size 5 and 10 negatives.

We relied on recent work on natural language inference to choose how to combine the student's answers and correct answer representations into a feature vector. Indeed, our task could be cast in a manner similar to inference, as the student's answer should be entailed by the correct answer to obtain a pass judgement. Therefore, we opted for a concatenation of the embeddings of the premise and hypothesis [11,32], which has been proven effective for the task of textual inference. Because each sentence is represented as embeddings of 300 dimensions, the result obtained through concatenation was 600-dimensional. This representation was then input directly to the classifier.

In addition to sentence embeddings, we extracted a set of seven distance-based features, which should capture the lexical and semantic similarity between the students' answers and correct answers. We preprocessed the answers by removing stopwords (e.g. articles or prepositions) and transcribing the mathematical notations into natural language (e.g., *">"* as *"maggiore"* (*greater*)). The text was then processed with the TINT NLP Suite for Italian [4] to obtain part-of-speech (PoS) tagging, lemmatisation, and affix recognition.

The output from TINT was then used to compute the following distance-based features:

- Token/Lemmas overlap: A feature representing the number of overlapping tokens/lemmas between the two sentences normalised by their length. This feature captures the lexical similarity between the two strings.
- Presence of negations: This feature indicates whether a content word is negated in one sentence and not in the other. For each sentence, we identify negations according to the *'NEG'* PoS tag or the affix *'a-'* or *'in-'* (e.g., *"indipendente"*), and then consider the first content word occurring after the negation. We extract two features, one for each sentence, and the values are normalised by their length.

Four additional distance-based features were computed using the sentence embeddings generated in the previous step and the single word embeddings, obtained again with fastText [10]:

- Cosine of sentence embeddings: We computed the cosine between the sentence embeddings of the students' answers and that of the correct answers. When represented in the same multidimensional space, the embeddings of two sentences with similar meanings were expected to be closer.
- Cosine of (lemmatised) sentence embeddings: the same feature as the previous item, with the only difference being that the sentences were first lemmatised before creating the embeddings.
- Word mover's distance (WMD): WMD is a similarity measure based on the minimum amount of distance that the embedded words of one document must move to match the embedded words of another document [35]. Compared with other existing similarity measures, it functions well when two sentences have a similar meaning despite having limited words in common. We applied this algorithm to measure the distance between the solutions proposed by the students and the correct solutions. Unlike the previous features, this measure was computed by considering the embedding of each word composing a sentence.
- WMD (lemmatised): The same feature as the previous item, with the only difference that the sentences were lemmatised before creating the embeddings.

In the classification experiments, we grouped and compared the features as follows:

- Sentence embeddings + distance features: We concatenated the 600 dimensional vector encoding the student and correct answer with the seven distance features described previously;
- Only sentence embeddings: the classification model was built using only the 600-dimensional vectors, without explicitly encoding the lexical and linguistic features of the student's comment and the correct answer.

As a baseline, we also computed the classification results obtained without sentence embeddings, using only the seven distance features.

4.3 Parameter setting

The classifier was implemented through a tuned SVM [47]. We initially found the best C and γ parameters using grid-search tuning [30], through a 10-fold cross-validation, to prevent over-fitting the model and better address the dataset size, which was not large. The best parameters were $C = 10^4$ and $\gamma = 2^{-6}$ for the complete setting (i.e., embedding + distance features). With the same approach, we also tuned the classifier when the input was only the concatenated sentence embeddings as features (i.e., without distance-based features), thus finding the best parameters of $C = 10^3$ and $\gamma = 2^{-3}$.

4.4 Results

4.4.1 Evaluation of answer classification

The tuned models presented in the previous subsection yielded the results summarised in Table 1.

	Embeddings and distance	*Embeddings only*
Accuracy	0.891	0.885
Balanced accuracy	0.876	0.870
F1 score	0.914	0.909
Cohen's K	0.764	0.752

Table 1 Accuracy, balanced accuracy, F1 score, and Cohen's K

The results indicate only a slight improvement in performance when using distance-based features in addition to sentence embeddings. This outcome highlights the effectiveness of using sentence embeddings to represent the semantic content of the answers in tasks where the students' answers and gold standards were similar. In fact, the sentence pairs in our dataset indicated a high level of word overlap, and the only discriminant between a correct and incorrect answer was sometimes only the presence of "$<$" instead of "$>$", or a negation. We manually inspected the misclassified and correctly classified pairs to verify whether there were common patterns shared by the two groups. From a surface point of view, no differences could be observed: the average sentence length, lexical overlap between pairs, and presence of negations were equally present in the misclassified and correctly classified pairs. The only remarkable difference was the presence, among false negatives, of pairs that belonged to the Pass class yet were manually graded as "partially correct", mainly because the student's

wording was not fully precise, whereas the professor could infer that the meaning of the answer was correct. Such cases were also not necessarily clear-cut for a human grader and mostly led the classifier into error.

The baseline classifier, obtained using only the seven distance-based features, yielded an F1 score of 0.758 for the Pass class and 0.035 for the Fail class, indicating that these features contribute to a superior assessment of matching pairs yet do not provide useful information concerning the other class. For comparison, we also experimented with feature vectors obtained using BERT [20], which processes words in relation to all the other words in a sentence, rather than one-by-one in order. BERT models can therefore judge the full context of a word by considering the words that precede and follow it. We tested these models because they have achieved state-of-the-art performance in natural language inference tasks [53], which inspired the choice to concatenate the students' answers and correct answers in the proposed approach. They were also successfully applied to short-answer grading on English data, with larger datasets than ours [39,50]. In our experiments, we adopted the Base Multilingual Cased model[2], covering 104 languages with 12 layers, 768 dimensional states, 12 heads, and 110M parameters. We fine-tuned the classifier using a maximum sequence length of 128, a batch size of 32, and a learning rate of 2^{-5}, executing it for five epochs. The model, however, did not converge, and no stable results were achieved, likely because of the limited dimension of the training set. We concluded that it could be possible to adopt this kind of transformer-based approach only if more training data were collected.

4.4.2 Comparison of manual vs automated grading

This subsection reports on the quality of the automated grading with respect to manual grading. Specifically, the automated grading tool was used to grade the solutions of a set of assignments used in one year of exams, from December 2018 to September 2019. The results were then compared with the grades assigned by the professor. The comparison consisted of:

- in terms of the numerical grade:
  - the intraclass correlation coefficient [6] to measure the level of agreement of the automated/manual grading;
  - the linear regression [52] to understand whether and how the manual grade could be predicted by the automated grade;
- in terms of a dichotomic fail/pass grade:
  - the Cohen's K [16] to measure the level of agreement of the automated/manual grading.

We analysed 122 solutions belonging to 11 different assignments, all with the same structure as the assignment in Section 2. The solutions were submitted in ten different assessment rounds. The average grade was 24/30 (s.d. 7); 18% of the solutions were considered unsatisfactory to pass the exam.

By comparing the manual and automated grades, the intraclass correlation coefficient was measured as 0.893, with 95% confidence intervals of $[0.850, 0.924]$. Such a value can be interpreted as an excellent indication of agreement between the two sets of grades [15].

Figure 3 displays the manual/automated grades and linear regression line with confidence intervals, where the automated grade is the independent variable and the manual grade is the dependent variable. The linear regression model was statistically significant ($p < 0.001$) and resulted in an $R^2 of 0.829$. This result indicates that the data were close to the

[2] https://github.com/google-research/bert/blob/master/multilingual.md

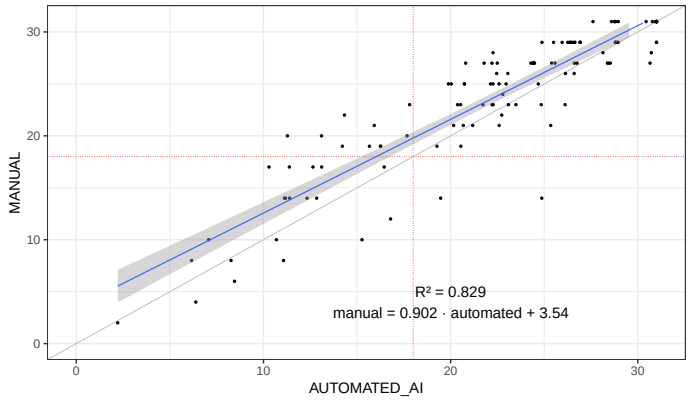

Fig. 3 Linear regression

regression line, and therefore, there was an excellent linear relationship between what was measured automatically and what was assessed by the professor. Furthermore, by observing the regression line (MANUAL $= 3.54 + 0.902 \cdot$ AUTOMATED), it is clear that the automated grading tool was more "conservative" than the professor, i.e., it returned lower grades on average. Finally, compared with the results achieved with the first implementation of the tool (see Section 3), we measured an increase of 0.089 points in R^2.

By transforming the numerical grades into dichotomic pass/fail outcomes, the agreement measured with Cohen's K was 0.71. This value is close to satisfactory. However, ten solutions that were considered insufficient and another two that were considered sufficient were instead considered as the opposite by the professor. These exams were the points in Figure 3 in the top-left and bottom-right portions of the Cartesian plane.

The final step performed was the calibration of the entire grading process, i.e., to determine the best values for the weights w_m, w_d, and w_c of Eq. (1) such that, on average, the automated grade equalled the manual grade. A maximum-likelihood estimation [46] was then performed, which returned the following values: $w_m = 0.050630$, $w_d = 0.014417$, and $w_c = 0.026797$. Using these weights, the prediction of the manual degree from the automated degree is depicted in Figure 4.

5 Formative assessment

Students were invited to participate voluntarily in the use of the system as a formative assessment tool during the course. A total of 36 students participated in the study.

Figure 5 displays the interface used by the students to proceed with the exercises. The dashboard indicates a summary of the reservations, the available exercises, and the results (Figure 5-a). From the dashboard, a student could open the list of available exercises through either the link "Go to test", or the menu (Figure 5-b). Upon clicking on the "Attend" button, the system displays the assignment (Figure 5-c): the student can read the exercise, copy/paste his/her analyses from R into the system, and then invoke the tool through the "Solve and suggest corrections" button. Figure 5-d indicates the feedback received by the system, with the correct commands and outputs (in green), correct commands with incorrect output (in blue), missing or incorrect commands (in red), and estimate of the final grade.

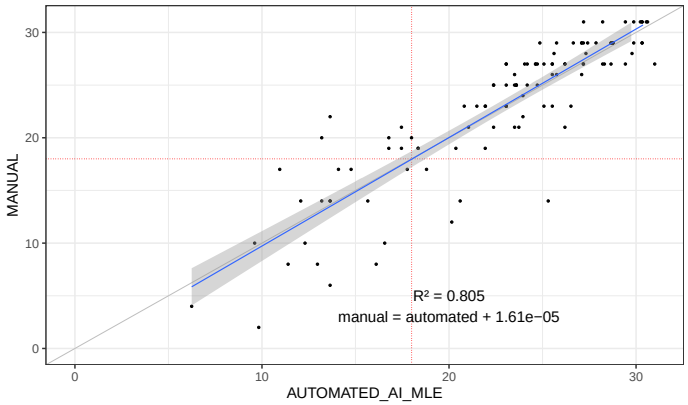

Fig. 4 Linear regression from maximum likelihood estimation

The screenshot regarding the exercise (i.e., Figure 5-c) displays, at the top, the text of the exercise; in the centre, an input area where the solution can be copied/pasted from R; on the right, a green question mark button that can be used to obtain help regarding the assignment; and at the bottom, three buttons that can be used to request feedback on the solution, save the solution, and go to the next part of the assignment. The screenshot in Figure 5-d) displays the feedback received by the tool (i.e., the textual notes and grade calculated as described in Section 3). In particular, the feedback reports that a normality test was calculated correctly (in green), another — although the command seemed correct — returned an incorrect value because either the data or the command syntax was incorrect, the t-test was not issued, and the estimated grade was 0.78 (ca 24/30).

To analyse the user experience with the formative assessment tool, we used the after scenario questionnaire (ASQ) [38]. The questionnaire was composed of three simple questions, i.e., Q1 = "Overall, I am satisfied with the ease of completing the tasks in this scenario", Q2 = "Overall, I am satisfied with the amount of time required to complete the tasks in this scenario", and Q3 = "Overall, I am satisfied with the support information (online help, messages, documentation) when completing the tasks". Each question allowed an answer from "1" to "7", where "1" indicated "Strongly disagree" and "7" indicated "Strongly agree". We opted for the ASQ because it touches upon the three fundamental areas of usability (i.e., effectiveness with Q1, efficiency with Q2, and satisfaction with Q1, Q2, and Q3) with only three questions, where the higher the rating, the better the perceived usability. The results did not highlight particular problems: on average, $\bar{Q}_1 = 5.2, \bar{Q}_2 = 5.2$, and $\bar{Q}_3 = 5.5$.

To complement these results, we asked the students how they used the tool. The answers indicated that they iteratively used the system to receive feedback and refine the solution until achieving the highest possible grade, verify the preparation for the final exam (62%), understand the correct statistical method to solve the problem (54%), and understand their mistakes (50%). The majority of the students (85%) asked to improve the feedback to better explain the mistake (i.e., why the analysis was not the correct one).

Figure 6 displays how students used the system with respect to the examination dates. Not surprisingly, most of the students used the system close to the examination date. In terms of time difference, calculated as the date of the exam minus the date of the exercise, we measured a mode of one day and a median of two days. Therefore, the majority of students used the system the day before the exam, with a central tendency of two days.

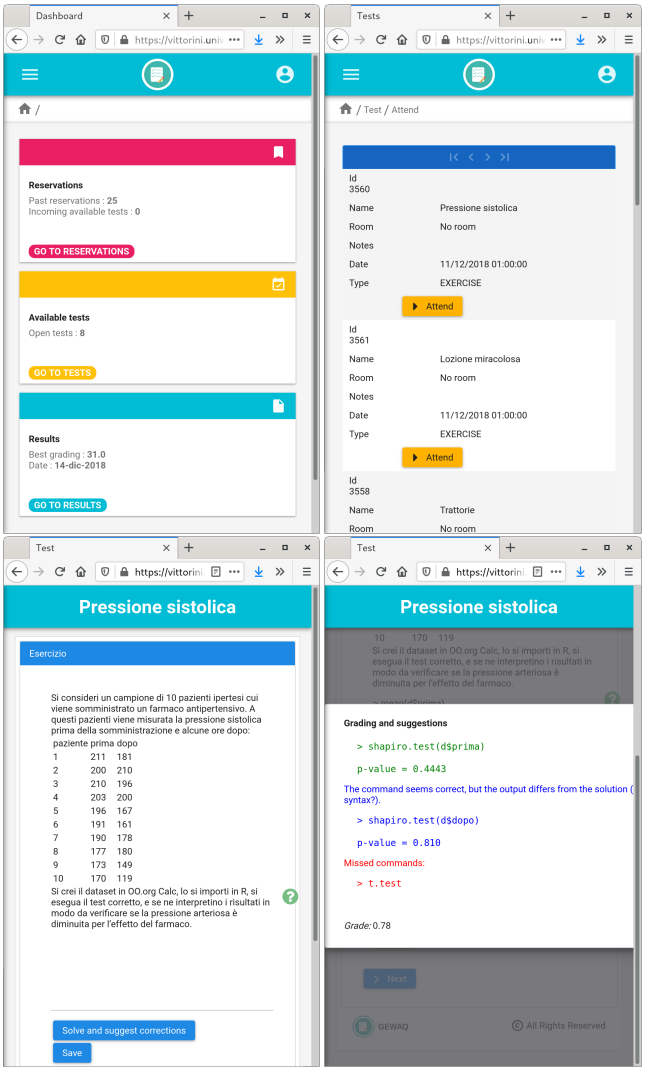

Fig. 5 User interface for students – formative assessment task. From top-left to bottom-right: (a) dashboard, (b) list of available exercises, (c) exercise, and (d) feedback. English translation in Appendix A

6 Summative assessment

As a summative assessment tool, in this section we report on the ability of the tool to increase the correction performance, support the professor in avoiding errors, and help students achieve improved grades.

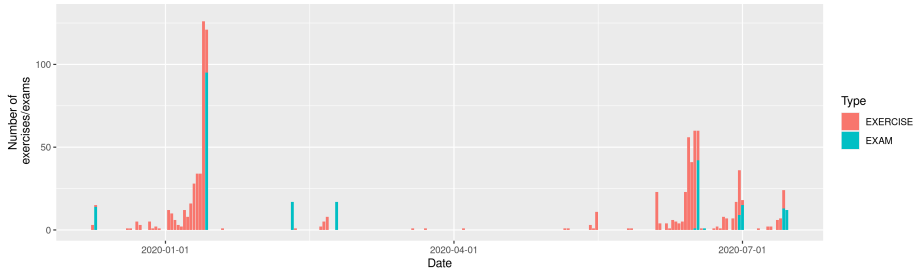

Fig. 6 Number of exercises/exams by date

6.1 Correction performances

Figure 7 depicts the interface that can be used by a professor to manually correct an assignment. The manual correction page displays a list of the available exercises (Figure 7-a), with information regarding the name of the exercise, submission date/time, student, grade, and two buttons: one to go to the correction page, the other to display the corrections in a pop-up dialog. By clicking on the first button, the correction page is displayed (Figure 7-b). As depicted in the figure, the interface presents (i) the name of the exercise, (ii) the name of the student, (iii) the text area where the professor can write his/her evaluation notes, (iv) the final grade, (v) a set of buttons to save the evaluation, collect the notes from the test sections, and send a notification email to the student, and (vi) the list sections of which the test is composed (in the figure, the actual exercise, ASQ, and engagement questionnaires) that can be opened by clicking on the section title. Once opened (see Figure 7-c), the system displays a text area where the professor can write notes regarding the test section, the section grade, and a button to save the corrections, followed by the list of questions of which the section is composed. The figure displays the question opened, and in particular, the part of the answer given by the student. By scrolling to the bottom (see Figure 7-d), the system displays a text area for writing notes, an input text for the question grade, and two buttons to save the evaluation and activate the automated tool. If activated, the notes and grade produced by the tool overwrite the text area and input text.

In our experiment, a professor was asked to correct all the assignments of an assessment session, randomly with or without the aid of the automated grading tool. There were 44 assignments in total: 21 were evaluated without the tool and 23 with the tool. We measured both the time (in seconds) to evaluate the assignment t and its normalisation in terms of the length of the assignment t_l, to consider the possible bias of the different solutions' lengths.

	Manual		*Automated*			
	N	*Mean (SD)*	*N*	*Mean (SD)*	*Reduction (%)*	*p-value*
t	21	148.08 (86.33)	23	84.61 (34.99)	43%	0.00008 **
t_l	21	86.77 (66.02)	23	58.05 (62.85)	33%	0.00508 *

Table 2 Summary of analysis of correction performances. Times in seconds. Normalised times in milliseconds

The results are summarised in Table 2. The time taken to evaluate an assignment, on average, reduced from approximately 148 s to 85 s, i.e., a 43% reduction. In a similar case for the normalised time, the reduction was marginally less (i.e., 33%). Both reductions were

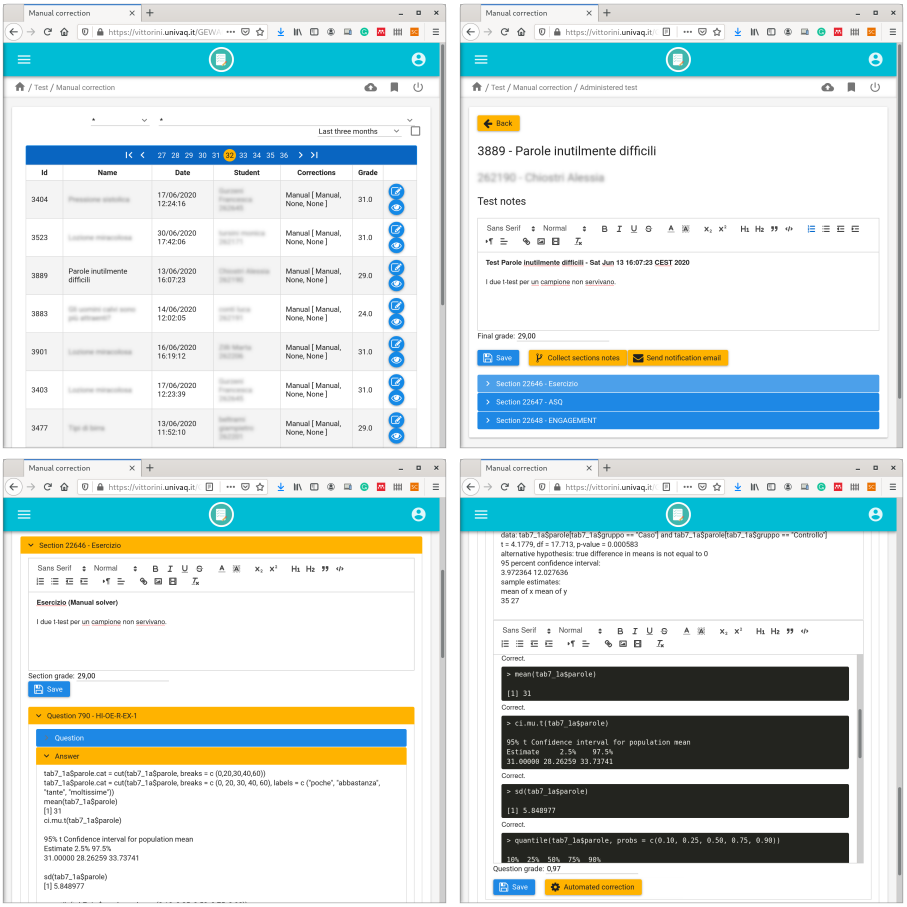

Fig. 7 User interface for professors – summative assessment task. From top-left to bottom-right: (a) list of exercises, (b) correction, (c) structure, and (d) automated correction. English translation in Appendix B

statistically significant. We used a Wilcoxon rank sum exact test to calculate the p-value because the distributions of t and t_l were not normal (verified with a Shapiro-Wilk test).

Finally, the 21 assignments manually evaluated by the professor were verified with the help of the automated tool; a further minor error was identified (a result was incorrect owing to a capitalisation error by the student).

6.2 Learning outcomes

The final research objective addressed whether students who used the tools for formative assessment demonstrated improved learning outcomes. With respect to students that did not use the system, Group 1, we compared the learning outcomes to those (i) who used the system for at least one exercise, (Group 2) and (ii) used the system for more than half of the proposed exercises, Group 3. We then measured the average degree obtained on the final exam (see the top of Table 3). The students from the first group achieved an average final degree of

21.7; the second group achieved 24.7; and the third 26.3. Both differences (i.e., Group 2 vs. Group 1 and Group 3 vs. Group 1) were statistically significant ($p = 0.0101$ and $p = 0.0146$). As with the performance, we used a Wilcoxon rank sum exact test to calculate the p-value because the distributions of the grades were not normal (verified with a Shapiro-Wilk test).

The higher grades of the students that used the system can be clearly mediated by other aspects such as previous knowledge of R and greater motivation. For the first confounding factor, given the fact that the students were from high school, it is quite unlikely that they could have developed a knowledge of R. For motivation, we measured the learners' engagement using the User Assessment Engagement Scale (UAES), which we had previously used and validated in [2]. The UAES comprised the following statements.

Focused attention	
UAES-FA-S1	I felt lost during the assessment (neg)
UAES-FA-S2	The time I spent during the assessment slipped away
UAES-FA-S3	I felt involved in the assessment
Perceived usability	
UAES-PU-S1	I felt frustrated during the assessment (neg)
UAES-PU-S2	I found the assessment confusing (neg)
UAES-PU-S3	The assessment performed in this fashion was taxing
Aesthetic appeal	
UAES-AE-S1	The system was attractive
UAES-AE-S2	The system was aesthetically appealing
UAES-AE-S3	The system was captivating
Rewarding	
UAES-RW-S1	Using the system was worthwhile
UAES-RW-S2	Performing the assessment in this fashion was rewarding
UAES-RW-S3	I felt interested in the assessment

It required a rating from "1" to "5", where 1 = Completely disagree, 2 = Agree, 3 = Neutral, 4 = Agree, and 5 = Completely agree. The questionnaire was scored as follows: (i) reverse the answers for items FA-S1, PU-S1, and PU-S2 and (ii) calculate the overall engagement score by adding all of the items together and dividing by twelve. Thus, in summary, the engagement was measured with a value ranging from "1" to "5", where the higher the value, the higher the engagement.

In the three groups (see bottom of Table 3), the small differences in terms of engagement did not increase with the number of completed exercises, and no differences were statistically significant. In this case, the distributions were normal, and we used a t-test. We also proceeded with a linear regression analysis, where engagement was the independent variable and grade was the dependent variable. The model did not fit ($p = 0.0649$), and the calculated $R^2 = 0.118$ implied an extremely low level of correlation between the two variables.

In summary, the two analyses suggest that the different grades achieved in the three groups did not appear to be mediated by a higher engagement. Nevertheless, it should be noted that only 24 students out of 125 (i.e., the 19.4%) actually completed the UAES, and therefore the results are limited only to this small sample.

7 Discussion and limitations

The automated grading provided by the tool in its entirety (i.e., code analysis and short-answer classification), refined using maximum likelihood estimation, demonstrated an excellent correlation with the human grading, even if specific solutions were incorrectly marked as

	Group	*N*	*Mean (SD)*	*p-value*
Grades	Group 1	43	21.7 (7.9)	
	Group 2	66	24.7 (6.1)	0.0101 *
	Group 3	15	26.3 (4.2)	0.0146 *
Engagement	Group 1	43	4.11 (0.4)	
	Group 2	66	4.29 (0.4)	0.3555 n.s.
	Group 3	15	4.17 (0.3)	0.8032 n.s.

Table 3 Learning outcomes: grades and engagement

fail/pass. Focusing on the short answers, representing them as embeddings and using them as features for a tuned SVM classifier yielded acceptable values of both F1 and Cohen's K. The addition of linguistically motivated features, capturing the distance between the short answers by the students and the gold standards did not substantially increase these results. This suggests that embeddings alone can capture the semantic similarity between two sentences even if their wording is different. The surprisingly good results confirm the effectiveness of sentence embeddings, in line with recent works in NLP. Nevertheless, this approach does have a limitation, i.e., the correct answers in the gold standard somewhat follow the same template, and the structure of well-formed students' answers was expected to match the same format. Indeed, during the course, the professor explained to the students how to understand the result of a test in terms of a process that (i) begins by finding the p-value in the R command output, (ii) then assesses whether the p-value is less than or greater than the threshold of 0.05, (iii) implying whether the null hypothesis must be rejected, and (iv) finally, providing an interpretation in terms of statistical and/or clinical meaning. This means that this structural pattern may play a role in discriminating between correct and incorrect answers of student, and that the presence of specific lexical cues signalling this structure could be more relevant than the semantic content itself.

The experiments conducted with the students demonstrated the following. As a formative assessment tool, students did not report any usability issues but reported its usefulness to the professor. As a summative assessment tool, the experiments indicated that the use of the tool increased the correction performance and helped the professor in avoiding errors. Furthermore, the students who used the system as a formative assessment tool had, on average, higher grades than the other students (and engagement did not appear to have any influence on such a difference). Conversely, there remain cases that suggest caution when using the tool as it can return false pass/fail outcomes, especially when close to the threshold.

8 Conclusions

The paper presented an automated grading tool that can support formative and summative assessment activities based on assignments containing both R commands and comments written in natural language. The paper initially focuses on the approach adopted to assess the answers provided by students, based on the concept of distance between a solution given by a student and the correct solution given by a professor. Then, it discusses the technical aspects of the tools, i.e., a static code analyser and supervised classifier relying on embeddings and distance-based features. The paper also reports on several experiments that suggest the usefulness of the tool for both students and professors.

The research discussed in this paper also opens opportunity for further work. The manner in which the students used the tool suggests possible improvements in how to structure the feedback [24], which will be implemented in the next release of the tool. Furthermore,

although the tool was developed, tested, and deployed to grade assignments regarding the statistical analysis of health data in R, the approach is general and could be applied to other domains and languages, provided a professor can collect a list of sentence pairs from previous exams, as each specific topic would require a set of these to tune the classifier [42]. Finally, the continuous use of the tool with students and the consequent collection of further correct/incorrect answers could also open the possibility of adopting transformer-based approaches that, to date, have not produced better results.

A — Exercise: Systolic Pressure

Below is the translation of the Italian texts in Figure 5.

Consider a sample of ten patients who received an antihypertensive drug. We measured the systolic blood pressure before and a number of hours after the administration:

Patient	Before	After
1	211	181
2	200	210
3	210	196
4	203	200
5	196	167
6	191	161
7	190	178
8	177	180
9	173	149
10	170	119

Create the dataset in OO.org Calc, import it into R, run the correct test, and interpret the results to verify whether the blood pressure decreased owing to the effect of the drug.

B — Correction notes

Below is the translation of the Italian texts in Figure 7.

Exercise title: Unnecessarily difficult words
Correction notes: One hypothesis testing is sufficient to solve the last point of the assignment.
Test section: Section 22646 - Exercise

References

1. Angelone, A.M., Menini, S., Tonelli, S., Vittorini, P.: Short sentences on R analyses in a health informatics subject (2019). DOI 10.5281/ZENODO.3257363
2. Angelone, A.M., Vittorini, P.: A Report on the Application of Adaptive Testing in a First Year University Course. In: Communications in Computer and Information Science, vol. 1011, pp. 439–449. Springer Verlag (2019). DOI 10.1007/978-3-030-20798-4{_}38
3. Angelone, A.M., Vittorini, P.: The Automated Grading of R Code Snippets: Preliminary Results in a Course of Health Informatics. In: Proc. of the 9th International Conference in Methodologies and Intelligent Systems for Technology Enhanced Learning. Springer (2019)
4. Aprosio, A.P., Moretti, G.: Tint 2.0: an All-inclusive Suite for NLP in Italian. In: Proceedings of the Fifth Italian Conference on Computational Linguistics (CLiC-it 2018). Torino (2018). URL http://ceur-ws.org/Vol-2253/paper58.pdf
5. Balfour, S.P.: Assessing Writing in MOOCs: Automated Essay Scoring and Calibrated Peer Review™. Research & Practice in Assessment **8**, 40–48 (2013)
6. Bartko, J.J.: The Intraclass Correlation Coefficient as a Measure of Reliability. Psychological Reports **19**(1), 3–11 (1966). DOI 10.2466/pr0.1966.19.1.3. URL http://www.ncbi.nlm.nih.gov/pubmed/5942109http://journals.sagepub.com/doi/10.2466/pr0.1966.19.1.3
7. Bernardi, A., Innamorati, C., Padovani, C., Romanelli, R., Saggino, A., Tommasi, M., Vittorini, P.: On the design and development of an assessment system with adaptive capabilities. In: Advances in Intelligent Systems and Computing, vol. 804. Springer, Cham (2019). DOI 10.1007/978-3-319-98872-6{_}23
8. Blumenstein, M., Green, S., Nguyen, A., Muthukkumarasamy, V.: GAME: A generic automated marking environment for programming assessment. In: International Conference on Information Technology: Coding Computing, ITCC, vol. 1, pp. 212–216 (2004). DOI 10.1109/ITCC.2004.1286454
9. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. arXiv preprint arXiv:1607.04606 (2016)
10. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics **5**, 135–146 (2017). DOI 10.1162/tacl_a_00051. URL https://www.aclweb.org/anthology/Q17-1010
11. Bowman, S.R., Angeli, G., Potts, C., Manning, C.D.: A large annotated corpus for learning natural language inference. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 632–642. Association for Computational Linguistics, Lisbon, Portugal (2015). DOI 10.18653/v1/D15-1075. URL https://www.aclweb.org/anthology/D15-1075
12. Burrows, S., Gurevych, I., Stein, B.: The eras and trends of automatic short answer grading. International Journal of Artificial Intelligence in Education **25**(1), 60–117 (2015)
13. Camus, L., Filighera, A.: Investigating transformers for automatic short answer grading. In: I.I. Bittencourt, M. Cukurova, K. Muldner, R. Luckin, E. Millán (eds.) Artificial Intelligence in Education, pp. 43–48. Springer International Publishing, Cham (2020)
14. Cheang, B., Kurnia, A., Lim, A., Oon, W.C.: On automated grading of programming assignments in an academic institution. Computers & Education **41**(2), 121–131 (2003)
15. Cicchetti, D.V.: Guidelines, Criteria, and Rules of Thumb for Evaluating Normed and Standardized Assessment Instruments in Psychology. Psychological Assessment **6**(4), 284–290 (1994)
16. Cohen, J.: A Coefficient of Agreement for Nominal Scales. Educational and Psychological Measurement **20**(1), 37–46 (1960). DOI 10.1177/001316446002000104. URL http://journals.sagepub.com/doi/10.1177/001316446002000104
17. Dawson-Howe, K.M.: Automatic Submission and Administration of Programming Assignments. ACM SIGCSE Bulletin **27**(4), 51–53 (1995). DOI 10.1145/216511.216539. URL http://portal.acm.org/citation.cfm?doid=216511.216539
18. De Gasperis, G., Menini, S., Tonelli, S., Vittorini, P.: Automated Grading Of Short Text Answers: Preliminary Results In A Course Of Health Informatics. In: ICWL 2019 : 18th International Conference on Web-Based Learning. Springer. LNCS., Magdeburg (2019)
19. Derval, G., Gego, A., Reinbold, P., Frantzen, B., Van Roy, P.: Automatic grading of programming exercises in a MOOC using the INGInious platform. European Stakeholder Summit on experiences and best practices in and around MOOCs (EMOOCS'15) pp. 86–91 (2015)
20. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (2019). DOI 10.18653/v1/N19-1423. URL https://www.aclweb.org/anthology/N19-1423

21. Edwards, S.H., Perez-Quinones, M.A.: Web-CAT: automatically grading programming assignments. In: Proceedings of the 13th annual conference on Innovation and technology in computer science education - ITiCSE '08, vol. 40, p. 328. ACM Press, New York, New York, USA (2008). DOI 10.1145/1384271.1384371. URL http://portal.acm.org/citation.cfm?doid=1384271.1384371
22. Fleming, W., Redish, K., Smyth, W.: Comparison of manual and automated marking of student programs. Information and Software Technology **30**(9), 547–552 (1988). DOI 10.1016/0950-5849(88)90133-4
23. Fox, A., Patterson, D., Joseph, S., McCulloch Paul: MAGIC: Massive Automated Grading in the Cloud. In: EC-TEL-WS 2015, Trends in Digital Education: Selected papers from EC-TEL 2015 Workshops CHANGEE, WAPLA, and HybridEd (2015). URL http://ceur-ws.org/Vol-1599/
24. Galassi, A., Vittorini, P.: Improved feedback in automated grading of data science assignments. In: Advances in Intelligent Systems and Computing, vol. 1236 AISC, pp. 296–300. Springer (2021). DOI 10.1007/978-3-030-52287-2{_}31. URL https://link.springer.com/chapter/10.1007/978-3-030-52287-2_31
25. Georgouli, K., Guerreiro, P.: Incorporating an Automatic Judge into Blended Learning Programming Activities. In: Advances in Web-Based Learning – ICWL 2010, pp. 81–90. Springer, Berlin, Heidelberg (2010). DOI 10.1007/978-3-642-17407-0{_}9. URL http://link.springer.com/10.1007/978-3-642-17407-0_9
26. Gomaa, W.H., Fahmy, A.A.: A Survey of Text Similarity Approaches. International Journal of Computer Applications **68**(13), 13–18 (2013). DOI 10.5120/11638-7118
27. Grave, E., Bojanowski, P., Gupta, P., Joulin, A., Mikolov, T.: Learning word vectors for 157 languages. In: Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018) (2018)
28. Harlen, W., James, M.: Assessment and Learning: differences and relationships between formative and summative assessment. Assessment in Education: Principles, Policy & Practice **4**(3), 365–379 (1997). DOI 10.1080/0969594970040304
29. Hollingsworth, J.: Automatic graders for programming classes. Communications of the ACM **3**(10), 528–529 (1960). DOI 10.1145/367415.367422. URL http://portal.acm.org/citation.cfm?doid=367415.367422
30. Hsu, C.W., Chang, C.C., Lin, C.J.: A Practical Guide to Support Vector Classification. Tech. rep., National Taiwan University (2016)
31. Jackson, D.: A semi-automated approach to online assessment. In: Proceedings of the 5th annual SIGCSE/SIGCUE ITiCSEconference on Innovation and technology in computer science education - ITiCSE '00, pp. 164–167. ACM Press, New York, New York, USA (2000). DOI 10.1145/343048.343160. URL http://portal.acm.org/citation.cfm?doid=343048.343160
32. Kiros, J., Chan, W.: Inferlite: Simple universal sentence representations from natural language inference data. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018, pp. 4868–4874 (2018). URL https://aclanthology.info/papers/D18-1524/d18-1524
33. Kuhn, M.: Building Predictive Models in R Using the caret Package. Journal of Statistical Software **28**(5), 1–26 (2008). DOI 10.18637/jss.v028.i05. URL http://www.jstatsoft.org/v28/i05/
34. Kurnia, A., Lim, A., Cheang, B.: Online Judge. Computers & Education **36**(4), 299–315 (2001). DOI 10.1016/s0360-1315(01)00018-5
35. Kusner, M., Sun, Y., Kolkin, N., Weinberger, K.: From word embeddings to document distances. In: International Conference on Machine Learning, pp. 957–966 (2015)
36. LeCounte, J.F., Johnson, D.: The MOOCs: Characteristics, Benefits, and Challenges to Both Industry and Higher Education. In: Handbook of Research on Innovative Technology Integration in Higher Education. IGI Global (2015)
37. Levenshtein, V.I.: Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Soviet Physics Doklady, Vol. 10, p.707 **10**, 707 (1966)
38. Lewis, J.R.: Psychometric evaluation of an after-scenario questionnaire for computer usability studies. ACM SIGCHI Bulletin **23**(1), 78–81 (1990). DOI 10.1145/122672.122692. URL http://portal.acm.org/citation.cfm?doid=122672.122692
39. Liu, T., Ding, W., Wang, Z., Tang, J., Huang, G.Y., Liu, Z.: Automatic short answer grading via multiway attention networks. In: International Conference on Artificial Intelligence in Education, pp. 169–173. Springer (2019)
40. Luck, M., Joy, M.: Automatic submission in an evolutionary approach to computer science teaching. Computers and Education **25**(3), 105–111 (1995). DOI 10.1016/0360-1315(95)00056-9. URL https://linkinghub.elsevier.com/retrieve/pii/0360131595000569
41. Magooda, A.E., Zahran, M., Rashwan, M., Raafat, H., Fayek, M.: Vector based techniques for short answer grading. In: The Twenty-Ninth International Flairs Conference (2016)

42. Menini, S., Tonelli, S., Gasperis, G.D., Vittorini, P.: Automated short answer grading: A simple solution for a difficult task. In: R. Bernardi, R. Navigli, G. Semeraro (eds.) Proceedings of the Sixth Italian Conference on Computational Linguistics, Bari, Italy, November 13-15, 2019, *CEUR Workshop Proceedings*, vol. 2481. CEUR-WS.org (2019). URL http://ceur-ws.org/Vol-2481/paper48.pdf
43. Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F.: e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien (2019). URL https://CRAN.R-project.org/package=e1071
44. Mohler, M., Bunescu, R., Mihalcea, R.: 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 - Volume 1, HLT '11, pp. 752–762. Association for Computational Linguistics, Stroudsburg, PA, USA (2011). URL http://dl.acm.org/citation.cfm?id=2002472.2002568
45. R Core Team: R: A Language and Environment for Statistical Computing (2018). URL https://www.R-project.org/
46. Rossi, R.J.: Mathematical Statistics : An Introduction to Likelihood Based Inference. Wiley (2018)
47. Scholkopf, B., Smola, A.J.: Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press (2001)
48. Souza, D.M., Felizardo, K.R., Barbosa, E.F.: A Systematic Literature Review of Assessment Tools for Programming Assignments. In: 2016 IEEE 29th International Conference on Software Engineering Education and Training (CSEET), pp. 147–156. IEEE (2016). DOI 10.1109/CSEET.2016.48
49. Sultan, M.A., Salazar, C., Sumner, T.: Fast and easy short answer grading with high accuracy. In: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1070–1075 (2016)
50. Sung, C., Dhamecha, T., Saha, S., Ma, T., Reddy, V., Arora, R.: Pre-training BERT on domain resources for short answer grading. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 6071–6075. Association for Computational Linguistics, Hong Kong, China (2019). DOI 10.18653/v1/D19-1628. URL https://www.aclweb.org/anthology/D19-1628
51. Sung, C., Dhamecha, T.I., Mukhi, N.: Improving short answer grading using transformer-based pre-training. In: S. Isotani, E. Millán, A. Ogan, P. Hastings, B. McLaren, R. Luckin (eds.) Artificial Intelligence in Education, pp. 469–481. Springer International Publishing, Cham (2019)
52. Weisberg, S.: Applied linear regression. John Wiley & Sons Inc (2013). URL https://www.wiley.com/en-us/Applied+Linear+Regression%2C+4th+Edition-p-9781118386088
53. Zhang, Z., Wu, Y., Li, Z., He, S., Zhao, H., Zhou, X., Zhou, X.: I know what you want: Semantic learning for text comprehension. CoRR **abs/1809.02794** (2018). URL http://arxiv.org/abs/1809.02794