Computers & Education 140 (2019) 103609

Contents lists available at ScienceDirect

Computers & Education

journal homepage: www.elsevier.com/locate/compedu

# Automatic detection of inconsistencies between numerical scores and textual feedback in peer-assessment processes with machine learning

Juan Ramón Rico-Juan, Antonio-Javier Gallego, Jorge Calvo-Zaragoza*

*Universidad de Alicante, Departamento de Lenguajes y Sistemas Informáticos, Carretera San Vicente del Raspeig s/n, 03690 Alicante, Spain*

ARTICLE INFO

*Keywords:*
Peer assessment
Open-ended works
Computer-aided assessment
Machine learning
Natural language processing

ABSTRACT

The use of peer assessment for open-ended activities has advantages for both teachers and students. Teachers might reduce the workload of the correction process and students achieve a better understanding of the subject by evaluating the activities of their peers. In order to ease the process, it is advisable to provide the students with a rubric over which performing the assessment of their peers; however, restricting themselves to provide only numerical scores is detrimental, as it prevents providing valuable feedback to others peers. Since this assessment produces two modalities of the same evaluation, namely numerical score and textual feedback, it is possible to apply automatic techniques to detect inconsistencies in the evaluation, thus minimizing the teachers' workload for supervising the whole process. This paper proposes a machine learning approach for the detection of such inconsistencies. To this end, we consider two different approaches, each of which is tested with different algorithms, in order to both evaluate the approach itself and find appropriate models to make it successful. The experiments carried out with 4 groups of students and 2 types of activities show that the proposed approach is able to yield reliable results, thus representing a valuable approach for ensuring a fair operation of the peer assessment process.

## 1. Introduction

Quite often, teachers have to face overcrowded classrooms (Shin & Teichler, 2014), which limits the possibility of carrying out certain activities because of the heavy workload that they involve. The use of computer tools may alleviate teacher's workload when dealing with this situation. For example, closed-answer activities can be easily corrected automatically, since the teacher should only prepare questions and specify the expected answers. Nevertheless, restricting students to closed-answer activities can be detrimental, as benefits of open-ended works, such as stimulating the originality or practicing the wording, are not taken into account. In turn, such open-ended works might represent unmanageable correction workload for the teacher, especially in the aforementioned overcrowded classroom scenario.

A widely considered alternative to mitigate the correction workload is to resort to peer assessment (PA) among students (Kulkarni et al., 2013): students evaluate the work of their classmates from which an aggregate grade is obtained. Such paradigm is not only ideal for reducing the teacher's workload, but it also allows students to learn from alternative solutions to the same problems proposed by their peers (Nicol, Thomson, & Breslin, 2014). It is important to emphasize that PA by itself does not prevent the teacher

* Corresponding author.
*E-mail addresses:* juanramonrico@ua.es (J.R. Rico-Juan), jgallego@dlsi.ua.es (A.-J. Gallego), jcalvo@dlsi.ua.es (J. Calvo-Zaragoza).

https://doi.org/10.1016/j.compedu.2019.103609
Received 7 February 2019; Received in revised form 9 May 2019; Accepted 17 June 2019
Available online 21 June 2019

from being involved in the correction process, since he or she is eventually responsible for the students to get a fair grade. However, the fact of obtaining several evaluations of the same work allows the use of statistical tools that help the teacher, as for instance by taking care of only those works in which there is no consensus among assessors (Rico-Juan et al., 2018) and avoiding self-assessment (Falchikov & Goldfinch, 2000).

In order to facilitate the process, the teacher can provide a set of evaluation rules (rubric) (Panadero, Romero, & Strijbos, 2013), so that the PA process is not totally free but the students have to restrict themselves to evaluating specific aspects of the works, as well as being forced to provide a numerical score for each activity. However, it is interesting that the evaluation also includes a textual review that can serve as feedback to the students evaluated, as well as forcing the assessors to clarify the reasons why such numerical score is determined (Li et al., 2016).

Interestingly, this means that the evaluation of each activity yields a double evaluation, namely numerical and textual ones. Numerical values represent the score given to the section being evaluated — similar to a Likert scale – and the text expresses suggestions for improvement. This duality represents an interesting scenario to detect inconsistencies between both scores proposed by the assessor, as for instance assigning a low score when the textual feedback indicates that everything is correct or assigning a high score when the textual feedback includes several suggestions to improve. Detecting this type of inconsistencies is a key step to ensure a greater fairness in the process, yet doing it manually would represent a heavy workload for the teacher. That is why in this paper we propose a system to perform this detection automatically using machine learning systems.

In our work, we have evaluated several techniques that are commonly used in the natural language processing (NLP) area to perform opinion mining or sentiment analysis, with the aim of estimating which numerical score would correspond to a specific textual feedback. Recent advances in NLP techniques suggest that their application to textual answers in the PA process is promising (Young, Hazarika, Poria, & Cambria, 2018). Our experiments, based on two different activities and around 1000 revisions, show that the approach is promising, and with the use of appropriate models very low error rates are attained. Our approach is postulated as an interesting tool to help teachers in a PA process with overcrowded classrooms, making them only have to pay attention to certain evaluations — those in which our system predicts a very different value to the one proposed by the assessor.

The rest of the article is structured as follows: Section 2 contextualizes the current work; Section 3 explains the NLP-based approaches considered; Section 4 details the experimental setup; Section 5 presents the results obtained; Section 6 elaborates on the main outcomes of our work; and finally, Section 7 concludes this paper and introduces some avenues for future research.

## 2. Contextualization

In closed answer activities it is usual to present only one correct answer. This feature allows an automated correction in a relatively straightforward manner (Wang, Chang, & Li, 2008). Some examples of these tasks have been successfully put into practice in the field of programming courses (Ala-Mutka, 2005), algebra (Pacheco-Venegas, López, & Andrade-Aréchiga, 2015), or in any multiple-choice evaluation tests.

Activities of open-ended works do not have a predefined answer, and can generally admit many valid solutions. This is why their correction involves a significantly greater effort than the correction of closed-answer works, and it is not straightforward to resort to automatic correction technologies (Bennett, Steffen, Singley, Morley, & Jacquemin, 1997). Furthermore, when the teacher wants to provide feedback to students about their activities — which might help during the learning process — the whole effort becomes unmanageable in overcrowded classes (Kulkarni et al., 2013). Within this context, PA is typically considered as an option to reduce such correction workload. In this case, open-ended tasks are evaluated directly by other students, with some additional benefits for themselves such as knowing different solutions to the same problem (Panadero & Brown, 2017) or being provided by a set of timely and useful feedback from their peers (Mulder, Pearce, & Baik, 2014).

PA certainly makes it easier to correct open-ended works. However, in a classroom context, assessors are other students who may not have clear assessment criteria. In these situations, it is usual to provide a rubric as a guide to facilitate the evaluation of the work itself and to standardize the criteria (Anglin, Anglin, Schumann, & Kaliski, 2008). In addition, it has been reported that the use of rubrics has a positive influence in the students' learning process (Brookhart & Chen, 2015; Panadero & Jonsson, 2013). However, although using PA conducted by rubrics facilitates certain tasks, the teacher's involvement is still needed throughout the process, both in preparing the rubrics for evaluation and in monitoring deliveries to ensure that there is no fraud or correction errors.

There are some previous attempts that have exploited this PA scenario in order to reduce teacher's workload. *Moodle* includes a module to handle PA: works are uploaded to the platform, and each work is automatically assigned to a fixed number of assessors; after each assessor provides a numerical score, and then the final score is computed as the median. The work of Rico-Juan et al. (2018) presents a statistically-based methodology for evaluating both the students' activities submitted, as well as the quality of the work done by the assessors. In the work of Luaces et al. (Luaces, Díez, & Bahamonde, 2018), matrix factorization techniques are used to provide both consistent grades and feedback to the students, and at the same time reducing the burden of the students in the whole process.

Furthermore, nowadays it is becoming increasingly common to find publications exploring the possibility of applying machine learning (ML) techniques — an area of artificial intelligence that studies how computers can learn from data — in the educational context (Barnes et al., 2017). For example, to predict the academic success of students in introductory programming courses (Costa, Fonseca, Santana, de Araújo, & Rego, 2017), to predict whether students will successfully complete their college degree (Daud et al., 2017), or to predict the selection of courses for a student in higher education (Kardan, Sadeghi, Ghidary, & Sani, 2013). There has been attempts to build ML methods for the automatic correction of open-ended works with the use of NLP (Noorbehbahani & Kardan, 2011; Xiong, Litman, & Schunn, 2012). However, their behaviour is still far from being robust and reliable.

Our paper presents a tool to help during the PA process of open-ended works, with the aim of automatically detecting inconsistencies between the numerical score and the textual feedback provided by the assessor. Our methodology is based on the use of a fixed rubric, which includes several sections for an activity. Each section focuses on a specific part of the work, which must be filled in with a Likert level scale and an open-text field with the suggestions that the assessor considers (e.g. 'everything is correct', 'this should be improved', 'the answer is not complete', etc.). Given a corpus of pairs (numerical score, textual feedback), an ML-based NLP model should be able to learn the numerical score that should correspond to a specific textual feedback. Such model could be then used to detect inconsistencies, and make the teacher pay attention to and correct — if necessary — only those reviews.

## 3. Natural language processing approaches

In this paper we consider ML algorithms for NLP in the context of textual feedback provided during a PA process. In general, ML is based on using examples of the task to be solved along with their corresponding expected predictions (usually referred to as ground-truth data in the computer-science literature). Using ML is known to be beneficial when generalizing the performance of the system in different contexts and activities, as opposed to hand-crafted heuristics.

The techniques that we consider are widely used in other NLP areas such as sentiment analysis or opinion mining. In our case, we want to detect automatically whether a textual feedback corresponds to a good or bad opinion of the activity under evaluation. These techniques usually consider millions of examples to train the predictive systems. Here, we shall study the behaviour of these algorithms in a context of about a few thousand different words, yet applied to a restricted domain according to the activity in the PA process based on rubrics. To build the ground-truth data set with which to train the systems, we shall use the textual feedback provided during the review as input and the numerical score proposed by that assessor as output.

In order to facilitate the analysis task by the aforementioned techniques, it is common to represent words with unique identifiers (integers) to handle the text as a sequence of numbers. To make these identifiers actually useful for our system, a basic pre-processing of the original text is advisable to make the system more robust. The pre-processing involves steps such as uncapitalizing words, removing special characters or punctuation marks, and replacing original words by their lexemes (see Fig. 1 up).

The sentences to be processed by the system are reduced to a variable-length sequence of identifiers. Prediction systems require that the input consists of a vector of fixed dimensions (Duda, Hart, & Stork, 2001), so the previous situation represents an obstacle. To solve the arbitrary length of the input sentences, two approaches will be studied (see Fig. 2). The first one uses recurrent models — which are capable of processing sequences of variable size — to convert any input sequence into a vector of fixed dimensions, which must contain discriminative features for the task at issue. The second one uses a direct prediction with Deep Neural Networks (DNN), that are context-aware models capable of taking decisions based on the relationship among features — words in this case. As most of these models also require fixed-length input entries, we shall consider the *padding* trick in which all sequences are converted into a fixed size by either trimming the surplus words or by filling in the sentences with *null* words.

Once the NLP model has been properly trained, it can be used to predict the numerical score that should be associated with a textual feedback. However, this approach must be understood as an aid to the teacher, and not as a fully-autonomous system. When it comes to putting it into practice, if the proposed numerical score and the score predicted by the NLP model agree, then we may trust

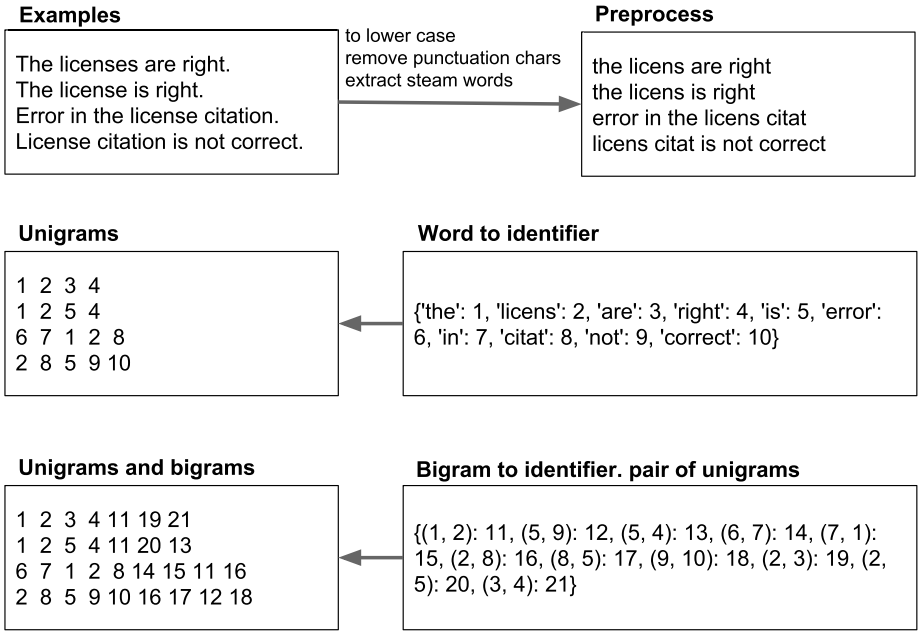



**Fig. 1.** A toy example with a few text samples preprocessed (up); unigrams extraction (middle); bigrams extraction (bottom).

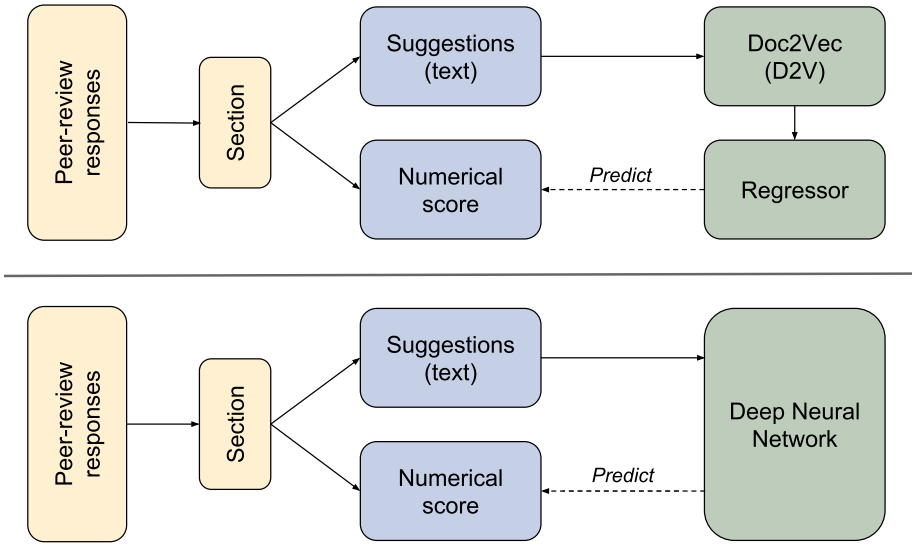



**Fig. 2.** Different approaches to predict the numerical score from the textual feedback. Up, two-step approach: from text to latent space and then prediction with a regression model. Down, one-step approach: from text to direct prediction using Deep Neural Networks.

such score. Otherwise, the teacher must be notified so as to manually verify those assessments where the inconsistency surpass a certain threshold. In the experimental section, we will study the impact of such threshold as regards the final score assigned to the student and the number of times the teacher is notified to manually review an assessment. In addition, this proposal is compared with two other approaches to deal with the same scenario.

Below, we explain in detail the two general NLP approaches considered to predict a numerical score from the textual feedback, along with the specific techniques chosen for performing such prediction.

### *3.1. Sentence embedding and prediction*

There are some algorithms like *word2vec* — created by a research group at Google and explained by other researchers (Goldberg & Levy, 2014) — or *Glove* (Pennington, Socher, & Manning, 2014), that generate a vector space where words that share a common context are close to each other. These so-called *embedding* vectors have advantages compared to earlier algorithms (Mikolov, Chen, Corrado, & Dean, 2013) such as latent semantic analysis (Dumais, 2004).

An evolved algorithm following the same idea is able to represent full pieces of text like sentences, paragraphs or documents onto a embedded numerical space. This algorithm is called *doc2vec* (Le & Mikolov, 2014) (D2V from now on). For this richer text representation, the embedding space has better performance than classic algorithms based on bag of words (Sinoara, Camacho-Collados, Rossi, Navigli, & Rezende, 2019). This makes it possible to tackle problems such as the direct classification of texts (Stein, Jaques, & Valiati, 2019) or sentiment analysis (Rezaeinia, Rahmani, Ghodsi, & Veisi, 2019) with more promising results.

The approach presented in this subsection is to apply D2V to each textual feedback of the PA in order to obtain a vector representation in such a restricted domain (all appearing words are closely related to a specific section of an activity). This representation will be used as input features of a prediction algorithm to estimate the expected score of that part of the activity (see upper part of Fig. 2).

In machine learning, algorithms that predict a continuous numerical value (rather than a discrete category) are known as regression models or regression systems. For this reason, we have selected a series of regression algorithms based on different strategies to cover the greatest number of approaches to the problem and thus being able to decide which of them is more appropriate.

The algorithms considered for this approach are presented as follows, along with a brief description:

• *K-Nearest Neighbors (kNN)* (Cover & Hart, 1967): it computes a numerical value based on the $k$ (parameter) nearest samples of the training set. It interpolates the final prediction based on the proximity of that neighbors according to the Euclidean distance. In our case, the parameter $k$ was fixed to 3 due to its good results in preliminary experiments.

- *Support Vector Machine (SVM)* (Drucker, Burges, Kaufman, Smola, & Vapnik, 1997): these algorithms are divided into two steps: the first one is to project the original features onto another space in which the predictions become linearly separable (typically using radial or polynomial transformations), and the second step is to learn a prediction of the samples in such projected space with the aim of maximizing the margin to the decision boundaries.
- *Decision tree* (Breiman, 2017): this model predicts the value of a sample by learning simple decision rules in the form of a tree. The tree is built from the training data, considering just one feature per rule.
- *Random Forest* (Breiman, 2001): it builds multiple decision trees in order to combine all predictions for a more robust behaviour.

- *Neural Network (Multilayer Perceptron)* (Hinton, 1990): it is the traditional neural network where all its layers are fully connected to each other.

### 3.2. Direct prediction with DNN

Recently, DNN have improved the performance in difficult machine learning problems (Goodfellow, Bengio, Courville, & Bengio, 2016). In particular, different network architectures have been used for natural language processing problems such as sentiment analysis or opinion mining (Glorot, Bordes, & Bengio, 2011; dos SantosGatti, 2014).

When using DNN for NLP tasks, it is common to place an *embedding* as first layer. The embedding layer is expected to learn to map any word identifier onto a space in which words that are related — for the task at issue — become represented by neighbouring vectors. This first layer depends on parameters such as the number of unique words or the vocabulary used, the dimension of the target vectors, and the maximum length of the sequences to be processed.

In the case of these experiments, we shall not only consider sequences of unique words (unigrams) but also represent the input as a sequence of bigrams, for which each pair of consecutive words are grouped together to form a unique identifier (see Fig. 1 middle and bottom). This allows us to check whether grouping words helps to improve the predictive power.

Regardless of the use of unigrams or bigramas, different neural architectures were tested to determine their suitability for the problem at hand:

- *Long short-term memory (LSTM)*: This is a type of recurrent neural network unit (Hochreiter & Schmidhuber, 1997), which is mainly used in sequence analysis (Rumelhart, Hinton, & Williams, 1986) as well as in language modeling problems (Sundermeyer, Schlüter, & Ney, 2012).
- *LSTM with attention mechanisms (LSTM + att)*: The attention mechanism helps the neural network to learn which parts of the input should be weighted in each case, with the intention of both helping convergence during learning and attaining a better performance (Vaswani et al., 2017).
- *CNN + LSTM*: This combination aims to extract the most relevant features of the sequence with a first convolutional layer, and then process its sequence representation with an LSTM layer.

As introduced above, all these models are complemented by an embedding layer, which is placed before the rest of the layers.

## 4. Experimentation

The experiments were conducted on a set of data extracted from two different activities in introductory computer-science courses. The topics considered for each of the activities were:

- **Activity 1. Creative Commons licenses:** In this activity, students have to choose a topic and search for five images on the Internet that satisfy a series of requirements concerning licenses. For this reason, the rubric also contains five sections, each of them dedicated to compiling the level of resolution of the task. Each section contains a text field for the assessor to fill in suggestions or observations for improvement.
- **Activity 2. Webquest:** In this case, a topic must be chosen and the corresponding steps must be taken to create a webquest in a correct way so that the contents are well structured, easily navigable, the material used is correctly cited, and the credits are properly presented. The rubric for this activity contains seven sections with several numeric fields per section and a text field for suggestions similar to the previous activity.

The correction rubrics used in our experiments are based on those described in 4. However, we have extended them to allow for including the textual feedback required by our approach. We collected new data from four different groups, consisting of a total of 354 students who submitted 176 works and conducted 1,925 revisions. An overview of the test case is given in Table 1. Additionally, in appendix A a topology description is detailed (Topping, 1998).

The experiments as regards the automatic detection of inconsistencies were performed in two ways: the first considers a single model for all sections (denoted as “All” in the tables) of the same activity, and the second considers a separate model for each section of the rubric of each activity (denoted as “Sections”).

Concerning the statistics of the corpora of textual feedback, Table 2 shows a summary of the text length in words, unique words and number of samples per activity for both unigrams and bigrams. As regards the number of words per sentence, we also include some statistical metrics for the sake of the analysis.

**Table 1**
Statistics of the PA process considered as our test case.

| Activity | Graders | Works | Revisions | Rubric sections |
|---|---|---|---|---|
| 1. Creative Commons | 175 | 91 | 956 | 5 |
| 2. Webquest | 179 | 85 | 969 | 7 |

**Table 2**
Statistics on the corpora of textual feedback considered in our experiments.

| n-gram | Activity | Section | Unique words | #examples | Number of words per sentence | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | Avg(std) | Min | Q1 | Q2 | Q3 | Max |
| 1 | 1 | 1 | 614 | 956 | 5.9(8.5) | 0 | 1 | 2 | 9 | 78 |
| | | 2 | 567 | | 4.9(7.3) | 0 | 2 | 2 | 7 | 46 |
| | | 3 | 595 | | 4.9(7.6) | 0 | 2 | 2 | 7 | 52 |
| | | 4 | 554 | | 5.2(7.8) | 0 | 2 | 2 | 7 | 54 |
| | | 5 | 591 | | 5.3(7.4) | 0 | 2 | 2 | 8 | 46 |
| | | All | 1103 | 4780 | 5.2(7.7) | 0 | 0 | 2 | 7 | 78 |
| | 2 | 1 | 749 | 969 | 7.8(9.5) | 0 | 2 | 4 | 11 | 102 |
| | | 2 | 668 | | 7.3(8.3) | 0 | 2 | 4 | 10 | 65 |
| | | 3 | 764 | | 9.5(10.3) | 0 | 2 | 6 | 12 | 79 |
| | | 4 | 768 | | 10.2(11.1) | 0 | 3 | 7 | 14 | 109 |
| | | 5 | 680 | | 7.9(8.9) | 0 | 2 | 5 | 10 | 70 |
| | | 6 | 651 | | 6.4(7.3) | 0 | 2 | 3 | 9 | 52 |
| | | 7 | 675 | | 9.7(10.4) | 0 | 2 | 6 | 13 | 97 |
| | | All | 1653 | 6783 | 8.4(9.6) | 0 | 2 | 5 | 11 | 109 |
| 2 | 1 | 1 | 2985 | 956 | 11.1(16.8) | 0 | 2 | 3 | 17 | 155 |
| | | 2 | 2572 | | 9.1(14.2) | 0 | 3 | 3 | 13 | 91 |
| | | 3 | 2622 | | 9.1(14.9) | 0 | 3 | 3 | 13 | 103 |
| | | 4 | 2569 | | 9.8(15.3) | 0 | 3 | 3 | 13 | 107 |
| | | 5 | 2772 | | 10.0(14.6) | 0 | 3 | 3 | 15 | 91 |
| | | All | 6921 | 4780 | 9.8(15.2) | 0 | 0 | 3 | 13 | 155 |
| | 2 | 1 | 4048 | 969 | 14.7(19.0) | 0 | 4 | 7 | 21 | 203 |
| | | 2 | 3634 | | 13.6(16.5) | 0 | 3 | 7 | 19 | 129 |
| | | 3 | 4445 | | 18.0(20.6) | 0 | 3 | 11 | 23 | 157 |
| | | 4 | 4738 | | 19.4(22.2) | 0 | 5 | 13 | 27 | 217 |
| | | 5 | 3616 | | 14.8(17.9) | 0 | 3 | 9 | 19 | 139 |
| | | 6 | 3214 | | 11.9(14.5) | 0 | 3 | 5 | 17 | 103 |
| | | 7 | 3789 | | 18.4(20.8) | 0 | 3 | 11 | 25 | 193 |
| | | All | 14874 | 6783 | 15.8(19.1) | 0 | 3 | 9 | 21 | 217 |

We can see that the average lengths of words per sentence in activity 1 are approximately half than in activity 2. This may be caused because the second activity is more complex and the suggestions are therefore longer. The Q1 is zero, since it corresponds to empty feedback when the activity is totally correct — according to the assessor — and there is no need for suggestions or observations. We can observe very high values in the max field, because there are some textual feedback that is extremely verbose. When taking into account all sections of the activity, there is approximately twice as much vocabulary (unique words) as for the cases separated by sections.

A cross validation scheme with 10 folds (10-CV) was used. Mean absolute error (MAE) was used to measure the accuracy of the results obtained from the predictive algorithms. This metric was chosen because of its easy interpretation in this teaching context, as it represents the absolute difference between the automatic prediction and the actual value. Therefore, the lower the MAE, the better the model.

In the next section, we will evaluate the MAE obtained by the different approaches considered with respect to the numerical value expected for each textual feedback. This will allow us to find the best model to detect the inconsistent answers given by the students. In Section 5.2, we will discuss the effect of incorporating this method in a real PA case of study.

### *4.1. Experimental setup*

To obtain the results of each activity we initially fix a series of parameters for the models. The specific configuration of each model was experimentally determined, yet inspired by experiments proposed in similar tasks.

The D2V transformation is built using *gensim* toolkit [1] (v3.4), that implements the *distributed memory* approach described in the work of 28. We set the dimension of the embedding space to 10. All the regression algorithms introduced in Section 3.1 were implemented using the *Scikit-learn* package (v0.19) (Pedregosa et al., 2011). Default parameterization was considered, except for the kNN algorithm for which the *k* parameter was set to 3.

Concerning the deep models, let us recall that neural networks algorithms are tested using both unigrams and bigrams as inputs, and that we need the *padding* trick to make all sentences be of the same length. In the case of unigrams, the length was fixed to 80 words; in the case of bigrams, the length was fixed to 120 words. All our neural models consider a first embedding layer to map the unique word (unigram or bigram) identifiers onto a feature space. In both cases, the dimensionality of the feature space was fixed to 10.

[1] http://pypi.org/project/gensim/.

**Table 3**
Parameters of the DNN architectures used in the experiments. Embedding (*embedding_dim* = *d*) refers to an embedding layer that maps every input onto a projected space of *d* dimensions; Convolution1D (*filters* = *f*, *kernel_size* = *k*) denotes a 1D-convolution operator of *f* filters and kernel size of *k*; MaxPooling1D (*pool_size* = *p*) represents a down-sampling operation of the dominating value within a 1D-window of size *p*; LSTM(*n*) means a Long Short-Term Memory unit of *n* neurons; Dense(*n*) denotes a fully connected layer of *n* neurons; AtentionLayer () refers to the layer implementing an attention mechanism. The size of the vocabulary and the length of the input sequences are determined for each case according to Table 2.

| Network | Topology |
|---|---|
| LSTM | Embedding (embedding_dim = 10)<br>LSTM(64)<br>LSTM(64) |
| LSTM + att | Embedding (embedding_dim = 10)<br>AtentionLayer()<br>LSTM(64)<br>LSTM(64) |
| CNN + LSTM | Embedding (embedding_dim = 10)<br>Convolution1D (filters = 256, kernel_size = 3)<br>MaxPooling1D (pool_size = 4)<br>LSTM(64)<br>Dense(10) |

**Table 4**
Mean of MAE obtained after the application of the algorithms to **Activity 1** with the 10-CV technique. The best average results are marked in bold type. The smaller the number the better the result.

| Algorithm | | Rubric sections - Activity 1 | | | | | Avg(std) | All(std) |
|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | | |
| Doc2Vec + | kNN | 0.41 | 0.38 | 0.36 | 0.52 | 0.41 | 0.42(0.12) | 0.35(0.05) |
| | SVM | 0.42 | 0.37 | 0.39 | 0.52 | 0.44 | 0.43(0.17) | 0.42(0.07) |
| | Decision Tree | 0.47 | 0.44 | 0.45 | 0.57 | 0.47 | 0.48(0.11) | 0.47(0.04) |
| | Random Forest | 0.50 | 0.44 | 0.46 | 0.57 | 0.51 | 0.50(0.11) | 0.50(0.04) |
| | Neural Net | 0.47 | 0.42 | 0.44 | 0.53 | 0.49 | 0.47(0.10) | 0.45(0.03) |
| Embedding + | LSTM 1-g | 0.29 | 0.25 | 0.29 | 0.36 | 0.35 | 0.31(0.15) | **0.22**(0.06) |
| | LSTM + att 1-g | 0.35 | 0.31 | 0.35 | 0.39 | 0.41 | 0.36(0.18) | 0.26(0.07) |
| | CNN + LSTM 1-g | 0.50 | 0.45 | 0.47 | 0.51 | 0.55 | 0.50(0.13) | **0.25**(0.09) |
| | LSTM 2-g | 0.31 | 0.26 | 0.28 | 0.37 | 0.35 | 0.31(0.14) | **0.21**(0.05) |
| | LSTM + att 2-g | 0.33 | 0.33 | 0.33 | 0.41 | 0.40 | 0.36(0.17) | **0.23**(0.06) |
| | CNN + LSTM 2-g | 0.49 | 0.48 | 0.52 | 0.55 | 0.54 | 0.52(0.13) | 0.28(0.09) |
| | **Min** | 0.29 | 0.25 | 0.28 | 0.36 | 0.35 | 0.31 | 0.21 |
| | **Max** | 0.50 | 0.48 | 0.52 | 0.57 | 0.54 | 0.52 | 0.50 |

The deep neural models were implemented under Keras framework[2] (v2.2.4). The specific configuration of the architectures introduced in Section 3.2 is detailed in Table 3.

The learning of the network weights is performed for a maximum of 200 epochs using stochastic gradient descent (Bottou, 2010) with a mini-batch size of 8 samples, considering the adaptive learning rate proposed by Kingma and Ba (Adam) (Kingma & Ba, 2014) (with default parameterization). We also follow an *early stopping* strategy, and so the training process is stopped if the training loss does not decrease after 5 epochs.

## 5. Results

Our experiments are divided into two parts: our first experimentation reports the capability of each NLP approach to accurately predict a numerical score from a textual feedback. Then, we carry out a goal-directed evaluation with the most accurate model. This evaluation further elaborates on the integration of the proposed method and its impact in the implementation of a computer-aided PA process.

[2] https://keras.io/.

**Table 5**
Mean of MAE values obtained after the application of the algorithms to **activity 2** with the 10-CV technique. The best average results are marked in bold type. The smaller the number the better the result.

| Algorithm | | Rubric sections - Activity 2 | | | | | | | Avg(std) | All(std) |
|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | | |
| Doc2Vec + | kNN | 0.19 | 0.27 | 0.34 | 0.45 | 0.36 | 0.28 | 0.56 | 0.35(0.10) | 0.35(0.10) |
| | SVM | 0.21 | 0.26 | 0.32 | 0.43 | 0.38 | 0.28 | 0.57 | 0.35(0.12) | 0.37(0.12) |
| | Decision Tree | 0.23 | 0.31 | 0.36 | 0.49 | 0.41 | 0.33 | 0.60 | 0.39(0.10) | 0.39(0.09) |
| | Random Forest | 0.21 | 0.30 | 0.33 | 0.47 | 0.39 | 0.32 | 0.60 | 0.37(0.09) | 0.40(0.09) |
| | Neural Net | 0.24 | 0.30 | 0.34 | 0.47 | 0.39 | 0.35 | 0.59 | 0.38(0.10) | 0.40(0.09) |
| Embedding + | LSTM 1-g | 0.16 | 0.21 | 0.28 | 0.37 | 0.29 | 0.25 | 0.44 | 0.29(0.11) | **0.24**(0.13) |
| | LSTM + att 1-g | 0.15 | 0.22 | 0.28 | 0.41 | 0.31 | 0.25 | 0.47 | 0.30(0.12) | **0.25**(0.12) |
| | CNN + LSTM 1-g | 0.44 | 0.46 | 0.40 | 0.50 | 0.48 | 0.39 | 0.57 | 0.46(0.10) | 0.26(0.12) |
| | LSTM 2-g | 0.15 | 0.21 | 0.28 | 0.36 | 0.30 | 0.25 | 0.44 | 0.28(0.11) | **0.23**(0.12) |
| | LSTM + att 2-g | 0.16 | 0.23 | 0.28 | 0.38 | 0.33 | 0.25 | 0.46 | 0.30(0.12) | 0.26(0.12) |
| | CNN + LSTM 2-g | 0.41 | 0.39 | 0.46 | 0.49 | 0.41 | 0.49 | 0.52 | 0.45(0.11) | **0.25**(0.11) |
| | **Min** | 0.15 | 0.21 | 0.28 | 0.36 | 0.29 | 0.25 | 0.44 | 0.28 | 0.23 |
| | **Max** | 0.44 | 0.46 | 0.46 | 0.49 | 0.48 | 0.49 | 0.60 | 0.46 | 0.40 |

### 5.1. Evaluation of the score computed from the textual feedback

In this first experiment, we simply check which is the model that is capable of obtaining a more accurate prediction of the numerical value that corresponds to a certain textual feedback. The results of this experiment can be consulted in Table 4 (Activity 1) and Table 5 (Activity 2). It can be generally checked that both main approaches — the one based on the use of the *D2V* algorithm with a predictor and the one based on DNN — are able to learn from our data, as all results are quite fair.

The first thing we want to emphasize is that the choice of training the models with all the sections at once (column All) or for each section specifically (summarized in the column Avg) is dependent on the approach and activity considered. On the one hand, in activity 1 both approaches seem to prefer to train with all the sections at once, since results are generally better. However, the opposite phenomenon is reported in the case of activity 2 for the D2V approach.

If we look into the specific results with more detail, the approach based on DNN obtains generally better results in our experiments than the approach based on D2V, always attaining the lowest rates in both activities. It should be noted the use of LSTM with 2-g in this context, since it reports the most accurate behaviour with 0.22 and 0.23 MAE figures in activity 1 and 2, respectively. The use of a CNN combined with LSTM, however, seems to be detrimental for this task (0.25 and 0.25, at best).

In the case of the approaches based on D2V, the best results are obtained mainly by kNN, yet far from the best results: 0.35 in both activities. Since this algorithm is based on the proximity of feature vectors, its goodness points out to the fact that the *D2V* process actually groups the sentences with a similar meaning in a close region of the embedded space. The rest of the regression algorithms behave much worse, except for SVM (0.42 and 0.37) and Neural Net (0.45 and 0.40).

In spite of all the above, it is true that most of the reported values fall into narrow ranges of MAE, which might question the statistical significance of the differences among the considered models. In order to verify these results, the Wilcoxon test (Wilcoxon, 1945) was used to determine which results are significantly better with a certain confidence. Fig. 3 shows a pairwise comparison of statistical significance among all considered models for both activities. It can be checked that the aforementioned conclusions are confirmed here: (i) it is usually better to train the models joining the data from all sections, (ii) the approaches based on DNN are generally better than those based on D2V, and (iii) the LSTM with 2-g represents the best choice as regards the specific model.

### 5.2. Goal-directed evaluation

In this section we further analyze the results of the best model obtained in the previous experiment (LSTM 2-g). The purpose here is to analyze how similar is the score obtained automatically by the best approach with respect to the assessment made by the instructor (expert). For the sake of experimentation, all the works considered were also reviewed manually by a teacher to obtain the *real* score that should be assigned to each section of each activity. We can then perform a goal-directed evaluation of our approach, i.e., comparing the MAE of the evaluations obtained by PA with respect to those assigned by the teacher, as well as the number of times the teacher is required to verify manually a certain assessment. Note that, when the teacher manually corrects a certain assessment, the overall MAE is reduced — since the teacher assigns the note that must be assigned — at the expense of making more revisions.

For the sake of comparison, we also consider the methodology proposed in 4 for the same purpose, which is based on making teachers manually look into those students' assessments that are found to be statistical outliers. Additionally, we consider a *combined* methodology that makes the teacher assessment both the scores marked by our system and those marked by the above-mentioned work based on outliers. The comparison of final MAE and % of required manual reviews is illustrated in Fig. 4.

As can be seen with the new approaches, the error made regarding the score assigned by the expert is considerably reduced, being the combined approach the one that obtains the best results. This best result is obtained at the cost of some more effort for the teacher. However, the best results are obtained with a threshold between 0.5 and 1, without having to review many more scores. For

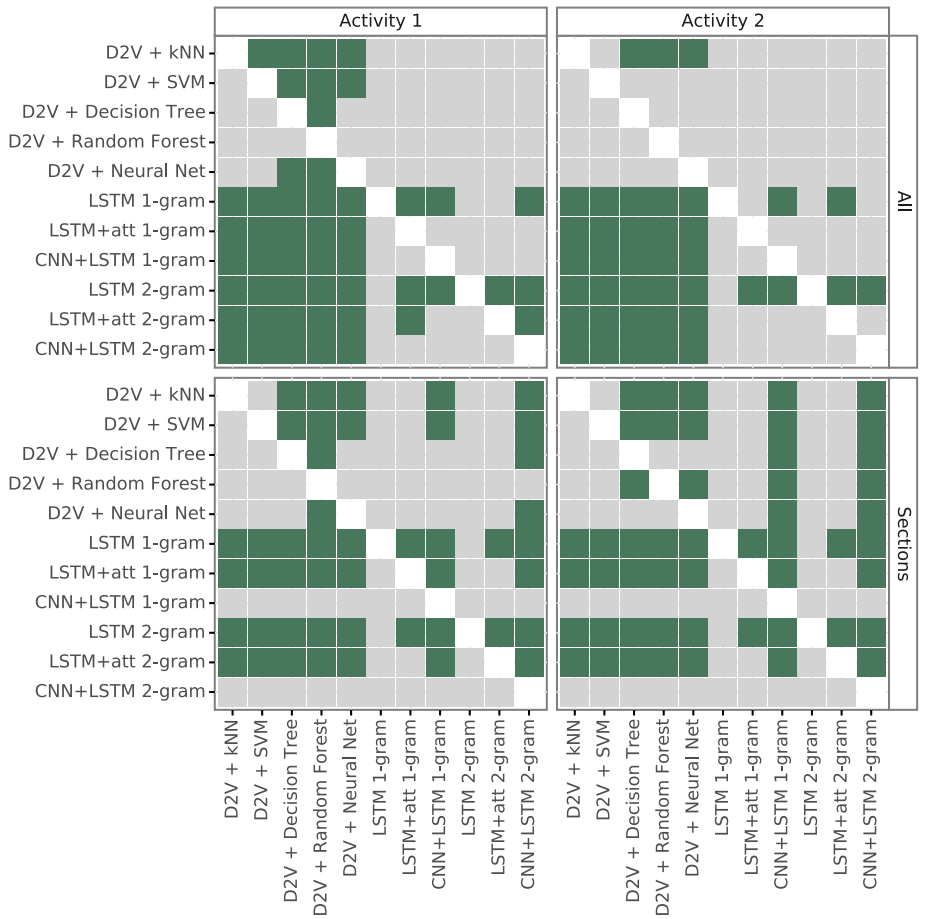



**Fig. 3.** Wilcoxon signed-rank test applied to algorithm pairs. The green (darker) cells mean that the algorithm of the row is significantly better (with a 95% of confidence) than the algorithm of the column. **All** represents a model training with all sections joined; **Sections** represents the average of training one model per section. (For interpretation of the references to colour in this figure legend, the reader is referred to the Web version of this article.)

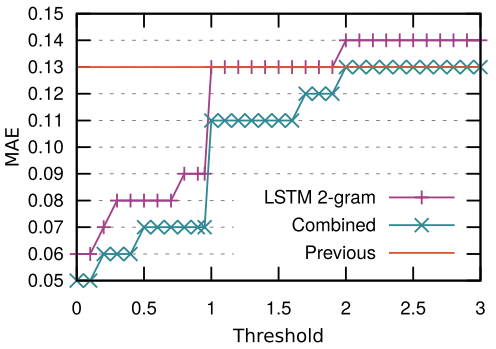
example, for activity 1 and a threshold of 0.5, the error is reduced by half (from 0.13 to 0.06) by checking less than 20% of the scores manually. For activity 2 and the same threshold, we managed to reduce it to less than half (from 0.16 to 0.07) but needing to review a little more (39%). In this case, the highest percentage of revisions may be due to the complexity of the activity, which has a greater number of subjective questions. In any case, these figures also show a relevant advantage of our approach: the flexibility it provides based on the threshold to be set.

In order to complete this analysis, Table 6 reports some examples of the discrepancies detected by the network for the two activities. In some cases, the textual feedback points out some important error of the activity but the student assigns the highest grade, whereas some other cases report a student saying that everything is correct but nonetheless assigns a low score.

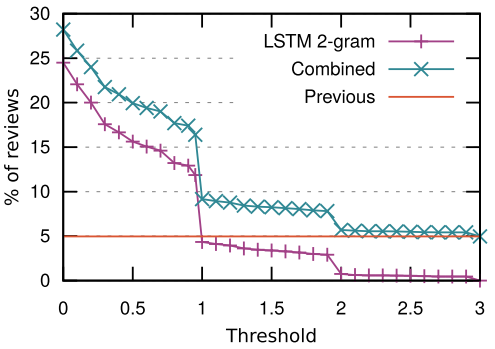
## 6. Discussion

As mentioned in previous sections, the proposed method represents a useful tool for performing the PA process in a semi-automatic way. Actually, the final evaluation is close to that of the teacher manually reviewing all the works, but reducing the workload to only review a small percentage of them (those whose revisions have been automatically marked as inconsistent). However, when taking our approach to a real scenario, a series of considerations must be first taken into account.

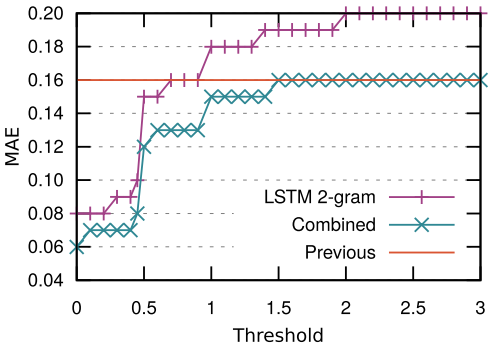
Concerning the PA configuration followed for our approach, it is widely known that it has clear advantages (Dochy, Segers, & Sluijsmans, 1999; Panadero, Jönsson, & Alqassab, 2018; Van Zundert, Sluijsmans, & Van Merriënboer, 2010). On the side of the students, they gain a formative experience by correcting other activities, since they can see other ways of solving the same problem and the main mistakes that other peers make. They also have to work with a rubric that provides the correction criteria, which makes them realize the important aspects to take into account. However, this type of PA approach is not suitable for all types of scenarios. It could be applicable only to those open-ended activities that can be assessed using a Likert scale and include a low percentage of subjective questions (as analyzed in Rico-Juan et al., 2018).



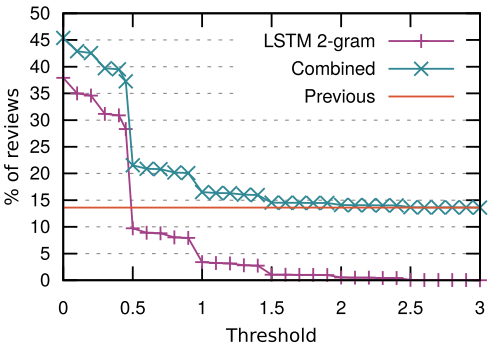

(a) Activity 1 – MAE




(b) Activity 1 – % of reviews




(c) Activity 2 – MAE




(d) Activity 2 – % of reviews

**Fig. 4.** Effect of the teacher's manual verification over those reviews detected by the approach presented in this paper, the approach proposed in 4, and a combination of both.

**Table 6**
Some examples of the discrepancies between the model prediction and the numerical score provided by the students for the two activities.

| | Num. score | LSTM 2-g | Student suggestion |
|---|---|---|---|
| Activity 1 | 3 | 0.6 | Everything is correct but as in the first image, the link does not let me access the image. |
| | 0 | 3 | Everything is correct/No suggestions. |
| | 1 | 3 | The image is suitable for the topic. |
| | 1 | 3 | I consider this is very complete and original. |
| | 3 | 1.1 | In my opinion, there is a lack of introduction explaining why the topic was chosen. The image itself is fine, but the license was CC-Attribution. The only condition was to reference the author and they have used an NC license as well. They wronged the license. |
| | 3 | 1.3 | The presentation is not that original. |
| | 1.3 | 3 | I think the presentation and the topic were fun and original. |
| | 1.3 | 3 | The image is nice, as well as the others, and very appropriate for the topic. |
| | 3 | 1.7 | The text reference was correct but not the graphic's one. |
| | 3 | 1.7 | The link is wrong, as in the others. |
| Activity 2 | 3 | 0.3 | There is a powerpoint that I cannot access. |
| | 3 | 0.5 | The links cannot be accessed directly. |
| | 3 | 0.5 | The bibliography is not included. |
| | 2.5 | 0.4 | The evaluation is not complete. |
| | 0 | 3 | No comments. |
| | 1 | 3 | Everything right. |
| | 3 | 1.5 | Missing bibliography. Missing author, topic, and level. |
| | 1.5 | 3 | The site is correct. |
| | 3 | 1.8 | The design is simple and there are no links to next pages. |
| | 2.5 | 1.4 | There is no information about the contents that students must learn. |

The teachers' workload reduction provided by implementing a PA system has been contested in empirical research, such in Panadero & Brown (Panadero & Brown, 2017), where the participants did not experience PA as saving their time. However, in this study there were no indications on how they implemented the PA, which is crucial to analyze the teachers' workload. In our study case, the time saving was significant, given that for the collection of assessments, online forms were used that automatically stored the results in a dataset. Thus, it became automatic to apply the proposed approach. However, a system of this type is not easy to put into practice (although once done, it can be reused). Therefore, another important factor to consider is the number of students over whom it will be applied, given that if they do not represent overcrowded classes—and if an online assessing system of this type is not available—it is possible that the temporary saving is not significant.

Since our method is based on machine learning, it has the advantages and disadvantages associated with this discipline: a wider generalization in the methodological aspect—the same approach can be easily reused—but it needs annotated data of the specific application domain at hand. However, there are ways to mitigate the latter, such as resorting to fine-tuning techniques: starting with a pre-trained model and adapting it with (few) data of the application topic. This workflow has been widely validated in the ML community, although it is difficult to calibrate *a priori* how much new data will be needed. Likewise, the required threshold to establish an inconsistency is hard to set uniquely for all possible scenarios; instead, we consider that the threshold could be dynamic, allowing the teacher to modify it according to the experience and needs.

Another important aspect to analyze about our proposal is the length of the comments given by the assessors. In our case of study, these comments were relatively short (see Table 2). This is mainly because we considered a simple evaluation rubric, which divided the activity into small parts for a more controlled scenario. Short texts usually express a single concept or idea, which should help to determine their positivity or negativity automatically. However, they also lack of contextual information and so their actual understanding is not straightforward.

It is worth mentioning that the literature reveals that there are many NLP approaches that work well for sentiment analysis applied to short phrases (such as tweets) with very good results, whose results are not that good with longer sentences (Arif, Li, Iqbal, & Liu, 2018; Jianqiang, Xiaolin, & Xuejun, 2018; Zhang, Wang, & Liu, 2018). For this reason, a crucial aspect in our proposal is to prepare a rubric that divides the assessment into small parts so as to ask questions about very specific aspects that can be answered shortly.

In addition, with the intention of obtaining good assessments and that students make an effort to assess correctly, it is also advisable to give a reward for carrying out the process correctly. In other words, not only assessing, but assessing correctly. In our case, 30% of the final grade of the student was granted for assessing correctly, given that, in addition, the proposed system allows to automatically detect when a assessment is not consistent or has assigned a score considered outlier according to the distribution of assessments made.

## 7. Conclusions and future works

In this work we assume a PA scenario for overcrowded classrooms, in which students evaluate their peers based on a series of sections of a rubric. The assessor is asked to assign a numerical score in each section, as well as a textual feedback that complements the decision made. For students, this procedure is beneficial since they acquire a greater knowledge about the activities because they have to evaluate those of their classmates, while generating valuable feedback. This scenario allows us to consider an automatic detector of inconsistencies between the numerical score and textual feedback provided by the assessors, via natural language processing techniques. The actual objective is to avoid that teachers have to look into every single evaluation of the PA, but just those in which inconsistencies are detected, thereby relieving their workload in the case of the aforementioned overcrowded classrooms.

Two different approaches have been presented in this paper to find inconsistencies. The first approach is based on finding a neutral embedded space for the textual answers that will be used as input for a regression algorithm that predicts the numerical score value associated with the processed text. In this case, *D2V* is used as an algorithm to embed the text in a vector space (also called latent space) that serves as input to several regressors belonging to different families of algorithms, that predict the final numerical score. The second approach performs the entire process using directly the text as input (sequence of words in the form of unigram or bigram) and the expected numerical score for the revised section as output, using different types of neural networks that have obtained very good results in similar tasks.

In the first case, kNN obtains the best results, which confirms the good performance of the *D2V* algorithm as the distribution of the embedded space is correctly used. But the second approach with recurrent networks as LSTM obtains the best global results with a MAE of 0.22 for activity 1 and 0.23 for activity 2. In addition, according to the significance tests, LSTM-based networks with 1-g or 2-g results are superior to the rest of the considered algorithms.

This paper shows how neural networks can be successfully used in restricted contexts (such as PA guided by a rubric) with a reasonable number of training samples (around a thousand per section) to detect inconsistencies between numerical and textual scores in the PA results. This process assists the teacher in his work of reviewing answers and helps him to focus attention on cases of inconsistency to take appropriate actions. In addition, the results obtained by applying the proposed methodology on different activities and several groups of students demonstrate how it is able to attain accurate results by reviewing only a low percentage of the works. This methodology is indeed similar to the case where the teacher reviews all the works manually, yet with a much lesser effort.

There might be some promising avenues for future work as regards technical issues, with which to improve the accuracy of the system. We believe, however, that the most promising idea is to shift the approach towards interactive learning. When the system detects an inconsistency, the teacher must manually check what happened. If it turns out that everything was correct — because the

system has found an inconsistency that is not such — the teacher could mark it, using this information to feedback the system so as to learn from human corrections.

Another interesting extension would be to integrate an online assistant into the form that would check the score and the feedback typed by the assessor to raise a warning when an inconsistence is detected. In this way, two advantages would obtained: the assessor would be able to double check the section before submitting his assessment, and the instructor would have fewer inconsistencies to review.

## Acknowledgment

The present work was supported through "Redes-I3CE de investigación en docencia universitaria" programme from "Instituto de Ciencias de la Educación de la Universidad de Alicante" (Ref.: REDES-I3CE-2018-4302)

## Appendix A. Pair assessment topology description

In peer assessment, activities do not exist in isolation. Peer evaluation is conducted in settings and activities that involve decisions regarding the use of peer assessment: the link between peer assessment and other elements of the learning environment, peer interaction, the composition of assessment groups, the management of the evaluation process, and contextual elements. Actually, there have been identified 17 different variables within the PA topology (Topping, 1998). Table 7 summarizes the descriptions of these variables for the experiments conducted in this study.

Table A.7
Description of the typology of Peer Assessment used in the experiments.

| # | Variable | Description |
|---|---|---|
| 1 | Curriculum subject | a) Creative Commons licenses; b) Webquest |
| 2 | Objectives | Staff saving time and students cognitive gains |
| 3 | Focus | Formative orientation |
| 4 | Product/output | a) Text/presentation; b) Website |
| 5 | Relation to staff assessment | A mixed system is used, It is substitutional but revised by the teacher in the cases of discrepancy between assessors or when inconsistencies in responses are detected. |
| 6 | Official weight | This is 70% of the activity grade of the assessed one, which receives about 10 evaluations. |
| 7 | Directionality | One-way |
| 8 | Privacy | The works assessed contain or do not contain the name of the authors according to their own decision. |
| 9 | Contact | The assessment is carried out from a distance using an online form. |
| 10 | Year | Same year of study. |
| 11 | Ability | The assessment is guided by a rubric to obtain maximum benefit from the skills of the assessor. |
| 12 | Constellation Assessors | Individuals |
| 13 | Constellation Assessed | The works sent were carried out in pairs or groups. |
| 14 | Place | In face-to-face class and staff is required |
| 15 | Time | Class time |
| 16 | Requirement | Voluntary for assessors |
| 17 | Reward | Correct evaluation contribute a 30% of assessor final grade. |

## References

Ala-Mutka, K. M. (2005). A survey of automated assessment approaches for programming assignments. *Computer Science Education, 15*, 83–102.
Anglin, L., Anglin, K., Schumann, P. L., & Kaliski, J. A. (2008). Improving the efficiency and effectiveness of grading through the use of computer-assisted grading rubrics. *Decision Sciences Journal of Innovative Education, 6*, 51–73.
Arif, M. H., Li, J., Iqbal, M., & Liu, K. (2018). Sentiment analysis and spam detection in short informal text using learning classifier systems. *Soft Computing, 22*, 7281–7291.
Rico-Juan, J. R., Gallego, A.-J., Valero-Mas, J. J., & Calvo-Zaragoza, J. (2018). Statistical semi-supervised system for grading multiple peer-reviewed open-ended works. *Computers & Education, 126*, 264–282.
Barnes, T., Boyer, K., Sharon, I., Hsiao, H., Le, N.-T., & Sosnovsky, S. (2017). Preface for the special issue on ai-supported education in computer science. *International Journal of Artificial Intelligence in Education, 27*, 1–4.
Bennett, R. E., Steffen, M., Singley, M. K., Morley, M., & Jacquemin, D. (1997). Evaluating an automatically scorable, open-ended response type for measuring mathematical reasoning in computer-adaptive tests. *Journal of Educational Measurement, 34*, 162–176.
Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. *Proceedings of COMPSTAT'2010* (pp. 177–186). Springer.
Breiman, L. (2001). Random forests. *Machine Learning, 45*, 5–32.
Breiman, L. (2017). *Classification and regression trees*. Routledge.
Brookhart, S. M., & Chen, F. (2015). The quality and effectiveness of descriptive rubrics. *Educational Review, 67*, 343–368.
Costa, E. B., Fonseca, B., Santana, M. A., de Araújo, F. F., & Rego, J. (2017). Evaluating the effectiveness of educational data mining techniques for early prediction of students' academic failure in introductory programming courses. *Computers in Human Behavior, 73*, 247–256.
Cover, T. M., & Hart, P. E. (1967). Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on, 13*, 21–27.
Daud, A., Aljohani, N. R., Abbasi, R. A., Lytras, M. D., Abbas, F., & Alowibdi, J. S. (2017). Predicting student performance using advanced learning analytics.

*Proceedings of the 26th international conference on world wide web companion* (pp. 415–421). International World Wide Web Conferences Steering Committee.
Dochy, F., Segers, M., & Sluijsmans, D. (1999). The use of self-, peer and co-assessment in higher education: A review. *Studies in Higher Education, 24*, 331–350.
Drucker, H., Burges, C. J., Kaufman, L., Smola, A. J., & Vapnik, V. (1997). Support vector regression machines. *Advances in neural information processing systems* (pp. 155–161).
Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern classification* (2nd ed.). Wiley.
Dumais, S. T. (2004). Latent semantic analysis. *Annual Review of Information Science and Technology, 38*, 188–230.
Falchikov, N., & Goldfinch, J. (2000). Student peer assessment in higher education: A meta-analysis comparing peer and teacher marks. *Review of Educational Research, 70*, 287–322.
Glorot, X., Bordes, A., & Bengio, Y. (2011). Domain adaptation for large-scale sentiment classification: A deep learning approach. *Proceedings of the 28th international conference on machine learning* (pp. 513–520). ICML-11.
Goldberg, Y., & Levy, O. (2014). *word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method.* arXiv:1402.3722, (p. [Online]).
Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning, Vol. 1*. Cambridge: MIT press.
Hinton, G. E. (1990). Connectionist learning procedures. *Machine learning: Vol. III*, (pp. 555–610). Elsevier.
Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*, 1735–1780.
Jianqiang, Z., Xiaolin, G., & Xuejun, Z. (2018). Deep convolution neural networks for twitter sentiment analysis. *IEEE Access, 6*, 23253–23260.
Kardan, A. A., Sadeghi, H., Ghidary, S. S., & Sani, M. R. F. (2013). Prediction of student course selection in online higher education institutes using neural network. *Computers & Education, 65*, 1–11.
Kingma, D. P., & Ba, J. (2014). *Adam: A method for stochastic optimization.* arXiv preprint arXiv:1412.6980.
Kulkarni, C., Wei, K. P., Le, H., Chia, D., Papadopoulos, K., Cheng, J., et al. (2013). Peer and self assessment in massive online classes. *ACM Transactions on Computer-Human Interaction, 20*, 1–31.
Le, Q. V., & Mikolov, T. (2014). Distributed representations of sentences and documents. *Proceedings of the 31th international conference on machine learning* (pp. 1188–1196).
Li, H., Xiong, Y., Zang, X., L. Kornhaber, M., Lyu, Y., Chung, K. S., & K. Suen, H. (2016). Peer assessment in the digital age: A meta-analysis comparing peer and teacher ratings. *Assessment & Evaluation in Higher Education, 41*, 245–264.
Luaces, O., Díez, J., & Bahamonde, A. (2018). A peer assessment method to provide feedback, consistent grading and reduce students' burden in massive teaching settings. *Computers & Education,* 283–295.
Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient estimation of word representations in vector space.* arXiv:1301.3781, (p. [Online]).
Mulder, R. A., Pearce, J. M., & Baik, C. (2014). Peer review in higher education: Student perceptions before and after participation. *Active Learning in Higher Education, 15*, 157–171.
Nicol, D., Thomson, A., & Breslin, C. (2014). Rethinking feedback practices in higher education: A peer review perspective. *Assessment & Evaluation in Higher Education, 39*, 102–122.
Noorbehbahani, F., & Kardan, A. A. (2011). The automatic assessment of free text answers using a modified BLEU algorithm. *Computers & Education, 56*, 337–345.
Pacheco-Venegas, N. D., López, G., & Andrade-Aréchiga, M. (2015). Conceptualization, development and implementation of a web-based system for automatic evaluation of mathematical expressions. *Computers & Education, 88*, 15–28.
Panadero, E., & Brown, G. T. (2017). Teachers' reasons for using peer assessment: Positive experience predicts use. *European Journal of Psychology of Education, 32*, 133–156.
Panadero, E., & Jonsson, A. (2013). The use of scoring rubrics for formative assessment purposes revisited: A review. *Educational Research Review, 9*, 129–144.
Panadero, E., Jönsson, A., & Alqassab, M. (2018). Providing formative peer feedback: What do we know? In A. A. Lipnevich, & J. K. Smith (Eds.). *The Cambridge handbook of instructional feedback*. Oxford: Cambridge University Press.
Panadero, E., Romero, M., & Strijbos, J.-W. (2013). The impact of a rubric and friendship on peer assessment: Effects on construct validity, performance, and perceptions of fairness and comfort. *Studies in Educational Evaluation, 39*, 195–203.
Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830.
Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. *Proceedings of the 2014 conference on empirical methods in natural language processing* (pp. 1532–1543). EMNLP).
Rezaeinia, S. M., Rahmani, R., Ghodsi, A., & Veisi, H. (2019). Sentiment analysis based on improved pre-trained word embeddings. *Expert Systems with Applications, 117*, 139–147.
Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning sequential structure in simple recurrent networks. *Parallel Distributed Processing: Experiments in the Microstructure of Cognition, 1*.
dos Santos, C., & Gatti, M. (2014). Deep convolutional neural networks for sentiment analysis of short texts. *Proceedings of COLING 2014, the 25th international conference on computational linguistics: Technical papers* (pp. 69–78).
Shin, J. C., & Teichler, U. (2014). The future of university in the post-massification era: A conceptual framework. *The future of the post-massified university at the crossroads: Restructuring systems and functions* (pp. 1–9). Cham: Springer International Publishing.
Sinoara, R. A., Camacho-Collados, J., Rossi, R. G., Navigli, R., & Rezende, S. O. (2019). Knowledge-enhanced document embeddings for text classification. *Knowledge-Based Systems, 163*, 955–971.
Stein, R. A., Jaques, P. A., & Valiati, J. F. (2019). An analysis of hierarchical text classification using word embeddings. *Information Sciences, 471*, 216–232.
Sundermeyer, M., Schlüter, R., & Ney, H. (2012). Lstm neural networks for language modeling. *Thirteenth annual conference of the international speech communication association*.
Topping, K. (1998). Peer assessment between students in colleges and universities. *Review of Educational Research, 68*, 249–276.
Van Zundert, M., Sluijsmans, D., & Van Merriënboer, J. (2010). Effective peer assessment processes: Research findings and future directions. *Learning and Instruction, 20*, 270–279.
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. *Advances in neural information processing systems* (pp. 5998–6008).
Wang, H.-C., Chang, C.-Y., & Li, T.-Y. (2008). Assessing creative problem-solving with automated text grading. *Computers & Education, 51*, 1450–1466.
Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin, 1*, 80–83.
Xiong, W., Litman, D., & Schunn, C. (2012). Natural Language Processing techniques for researching and improving peer feedback. *Journal of Writing Research, 4*, 155–176.
Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). Recent trends in deep learning based natural language processing [review article]. *IEEE Computational Intelligence Magazine, 13*, 55–75.
Zhang, L., Wang, S., & Liu, B. (2018). Deep learning for sentiment analysis: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 8*, e1253.