

An Integrated Learning System for First-Year University Students – with Automated Grading Modules for Typewriting, Microsoft Excel Spreadsheets and Word Documents –

Kazunori Iwata ^{*}, Yoshimitsu Matsui [†]

Abstract

This paper describes an integrated learning system for first-year students to learn basic computer skills, including automated grading modules for typewriting and MS-Excel files and MS-Word files. The system aims to relieve teachers' workloads to grade many MS-Excel and MS-Word files. It also provides immediate feedback and has a mechanism to prevent students from submitting copied files.

In addition, this paper describes the time to grade typewriting, MS-Excel, and MS-Word files. It computes the students' average normalized gain by using the operational records of the system in our university in 2021. The average normalized gain shows the variation between students' computer skills decreased. These results, therefore, indicate the effectiveness of the system.

Keywords: Automated Grading System, Integrated Learning System

1 Introduction

First-year university students need to acquire basic computer skills that are typewriting skills and using Microsoft Excel and Word (hereinafter referred to as MS-Excel and MS-Word).

Typewriting is a fundamental skill, and it is also one of the essential computer skills. Learning to type faster and accurately will help students in their university lives. MS-Excel and MS-Word are essential tools for university students to write reports and thesis. MS-Excel is used for calculations and data visualization. Students describe their opinions by using MS-Word.

Because of these, it is recommended that students at our university to take a course in typewriting, MS-Excel and MS-Word. This provides the students with typewriting exercises, basic skills in calculating data with functions, graphically presenting data, and formatting documents.

^{*}Faculty of Business Administration, Aichi University, Nagoya, Japan

[†]Faculty of Law, Aichi University, Nagoya, Japan

According to various learning theories, preparing practical exercises and providing timely feedback are crucial for successful learning. However, as the number of exercises in the course increase, teachers encounter problems such as follows:

- increased time needed to grade exercises
- difficulty in customizing feedback
- delays in providing feedback
- increased the number of copied files

Because of the above problems, we implemented an integrated learning system with automated grading modules for typewriting, MS-Excel, and MS-Word. The system can provide immediate feedback and has a mechanism to prevent students from submitting copied files.

In this paper, we describe the system in detail and demonstrate the performance of its modules. In addition, we show the operation records of the system in our university in 2021. The paper is organized as follows. Section 2 introduces the background and related works of this study. Section 3 illustrates our integrated learning system. Section 4 describes automated grading modules on the system. Section 5 shows experimental and operational results. Finally, Section 6 presents our conclusions.

This paper is an extension of work originally presented in 11th International Congress on Advanced Applied Informatics [1].

2 Background and Related Works

Our university students take introductory computer literacy courses. The number of students in this course is 1,500 per semester, and most of the students are inexperienced in using computers. Therefore, providing meaningful and numerous exercises and timely feedback on grading is a critical component of their successful learning [2].

Incidentally, ten teachers manage the courses in our university. Each teacher must check exercises from 150 students. If each teacher wants to provide 100 exercises for a student and needs 30 seconds to grade an exercise, each teacher consumes about 125 hours for 150 students only in the course. In that case, the student cannot receive timely feedback. We implemented an automated grading system for MS-Excel and MS-Word to solve the above problems.

There are many learning management systems (hereinafter referred to LMS), such as Moodle [3], Edmodo [4], and Blackboard [5]. However, these very smart LMSs do not provide automated grading systems.

Hekman [6] explained the related works in detail. Our system is similar to Kline and Janicki for grading MS-Excel files [7] but treats the formulae that show the same results is differently. The details are in subsection 4.2.1. In addition, our system also supports grading MS-Word files.

3 Implementation of an Integrated Learning System

This section explains our integrated learning system, called HITs (Highly Interactive Training System).

3.1 Outline of HITs

Figure 1 shows the outline of the system. It has three parts such as:

Interface: The interface works on the Apache HTTP Server and is implemented using HTML5, PHP, and JavaScript.

Database: The database stores users' information and records using PostgreSQL.

Grading modules: There are three grading modules for typewriting, MS-Excel and MS-Word. The module for MS-Word is written in Python, and the others are written in PHP. These modules run independently of the interface. In other words, they can easily be used on other systems.

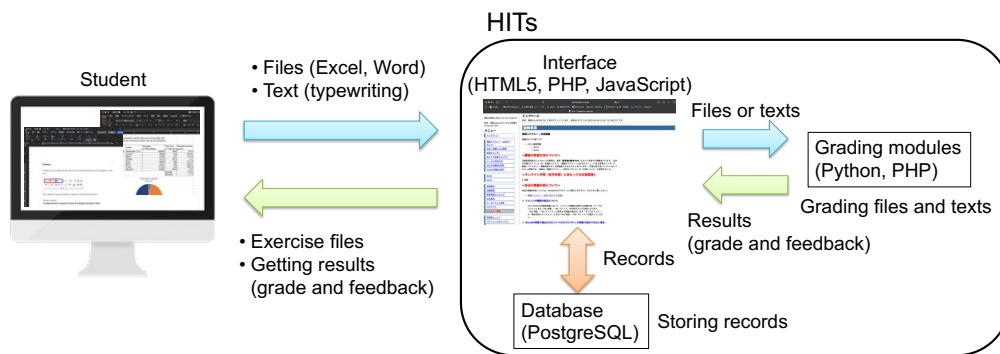


Figure 1: Outline of the system

The students' basic operations for MS-Excel and MS-Word are as follows:

1. Gain access to HITs via a Web browser.
2. Log in HITs.
3. Select an exercise (MS-Excel or MS-Word).
4. Download the exercise file.
5. Answer questions in the file by Personal Computer (PC).
6. Upload the file.
7. Receive the grading result and feedback.

If students select the typewriting exercises, they type texts on the Web browser.

3.2 Scoring Criteria

These modules grade students' files according to scoring criteria. The scoring criteria are called control files on HITs and are created from sample answer files (Figure 2). The control files are named after the exercise IDs, which are explained in subsection 3.3.

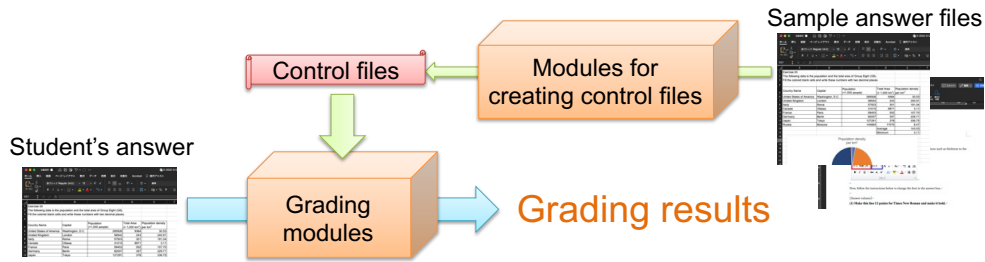


Figure 2: Scoring criteria

3.3 IDs of Exercises

The exercise IDs have the following format:

$$\alpha - x - y - z, \quad (1)$$

where $\alpha \in \{E, T, W\}$ and $x, y, z \in \mathbb{N} \cup \{0\}$.

The α indicates the exercise kind, E is MS-Excel, T is Typewriting, and W is MS-Word. The x can be used to control exercises in various ways. In our university, we use it to express the semester number. The y is the exercise managing number, which differs from the exercise number shown to students. The exercise numbers can be set independently.

The z shows a similar question number and is used if there are plural exercises for one exercise managing number y . For example, we set two similar MS-Excel questions as E-0-1-0 and E-0-1-1, respectively. The questions can be used to one bundle of question as Excel-1 and assigned to students separately. In other words, one student solves E-0-1-0, and another student solves E-0-1-1 as Excel-1. The number is not used to record a student's grade. Therefore, a student completing E-0-1-0 and another student completing E-0-1-0 have the same record that is they finish Excel-1.

3.4 Embedding Hash Values of Students' IDs

If teachers prepare numerous exercises for students, some students submit copied files. A question set using the exercise IDs is slightly practical but not crucial to avoid such plagiarism. Hence, HITs embeds student IDs and hash values into MS-Excel and MS-Word files when students download the files. The hash values are computed from student IDs, exercise IDs, and random numbers stored in the database. HITs verifies the hash values when students upload files (Figure 3). If a file fails the verification, a student receives a warning, and teachers receive the reports. The teachers judge whether the student submitted a copied file or not according to the reports and system logs.

3.5 Students' Ranking System

Cherry & Ellis [8] reported:

Findings suggest that student performance is significantly improved when facing a grading system based on student ranking (norm-reference grading) rather than performance standards (criterion-reference grading). The improved outcomes from rank-order grading largely arise among the high performers, but not at the expense of low performers.

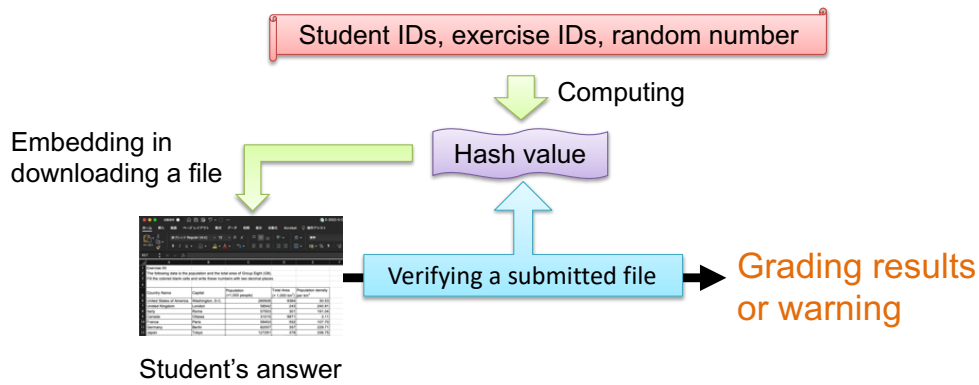


Figure 3: Embedding and verifying a hash value

Therefore, HITs have the students' ranking system according to the completion ratio of exercises. The ranking does not show the students' grading directly but encourages them to complete exercises. The example of the ranking is in Figure 4 ¹.

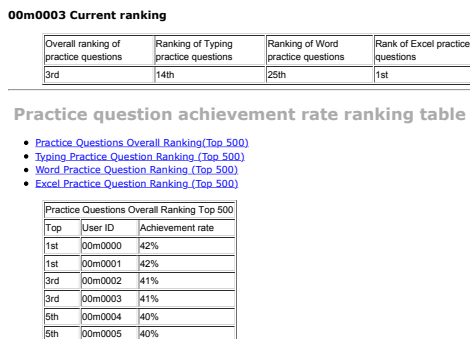


Figure 4: Example of the ranking system

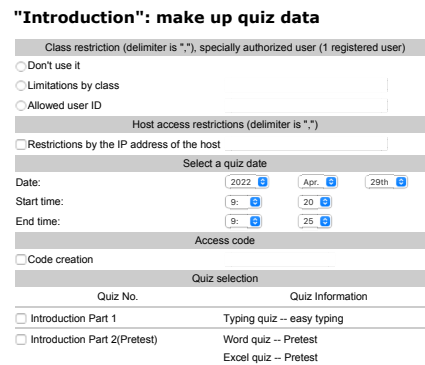


Figure 5: Example of quiz settings

3.6 Scheduling Control Functions for Exercises

HITs can specify times when exercises are accessible for students. The exercises are split into two parts. One is a practice, and the other is a quiz. Both parts have the following settings for the timing:

Open the exercise: The exercise will be unavailable to students before the opening time. Exercises with start times in the future display between the open and close times. If the open time does not exist, students can access an exercise anytime before a close time.

Close the exercise: The students will not be able to access a exercise after the closing time. Answers that the student submits after the exercise closing time will not be graded. If the close time does not exist, students will be able to access an exercise after an open time.

¹Almost all of the example figures of HITs are translated from Japanese to English by Google Chrome because our system currently supports only Japanese.

Restrict access for students: Timing may be changed for individual students or groups. If there is no restricted setting for an exercise, all users can access the exercise between the open and close times. An individual user setting gets preference over class settings.

Only the quiz part can use the following restrictions:

Access code: If the access code is specified, students must enter the same access code before they access the quiz.

Network address: HITs can restrict access for a quiz to particular networks on the LAN or Internet by specifying a comma-separated list of group or whole IP address numbers. This setting is used for proctoring a quiz, which only students in a specific room can access. There are two types of IP address numbers as follows:

- Group IP addresses, such as 192.168.1.* which will match computers from 192.168.1.1 to 192.168.1.255.
- Whole IP addresses, such as 192.168.1.1 which will match a single computer.

Figure 5 shows the example of the quiz part settings.

4 Automated Grading Modules

4.1 Grading Module for Typewriting

Figure 6 shows the interface of a typewriting exercise. Students type texts into the “Input area” to be identical to the text in “Problem statement”.

Enter the text shown below in the input area on the right.

<p>Problem statement:</p> <pre>jfjfjfjf hghghghg jfjfjfhghghghg</pre>	<p>Input area: elapsed time: 0</p> <div style="border: 1px solid gray; height: 40px; width: 100%;"></div> <p style="text-align: right;">How to use</p>
<p>Grading Clear Close</p>	

Figure 6: Typewriting interface

The grading module for typewriting has a simple feature to check students’ input. The module checks inputs after the student presses a “Grading” button after finishing an exercise. Most typewriting-test systems check inputs immediately after each character is entered. However, our module does not adopt that method because a kana-kanji conversion is important for inputting Japanese. In Japanese input, users type in phonetic “Hiragana,” but appropriate Japanese is written in logographic “Kanji.” A kana-kanji conversion substitutes a Kanji string for a Hiragana string. Standard kana-kanji conversion systems replace Hiragana with Kanji by pressing a space key.

The module can create immediate feedback for students using the O(ND) algorithm [9, 10], which quickly finds differences between a student’s input and a sample answer.

Control files for the module are merely text files that contain sample answers. They can include only the HTML tags “<ruby>,” “</ruby>,” “<rp>,” “</rp>,” “<rt>,” and “</rt>.” The tags are used to show the pronunciation (“Hiragana”) of a “Kanji” that might be hard to be read because a student needs to enter a Hiragana to input a Kanji.

4.2 Grading Module for MS-Excel

The grading module for an MS-Excel file can check the following items:

- value of a particular cell
- formatted value of a particular cell: decimal places, percentage style, and/or currency style
- data type in a particular cell: string, numeric, or formulae – a list of using functions
- existence of graphs
- type of graphs
- data range of graphs
- legends and axis labels of graphs

The module collects the items from a submitted an MS-Excel file by using PhpSpreadsheet [11].

In embedding a hash value, HITs locks cells that do not have scoring targets to protect them and colors scoring targets (Figure 7).

	A	B	C	D	E
1	Exercise 00				
2	The following data is the population and the total area of Group Eight (G8).				
3	Fill the colored blank cells and write these numbers with two decimal places.				
4					
5	Country Name	Capital	Population (×1,000 people)	Total Area (× 1,000 km ²)	Population density per km ²
6	United States of America	Washington, D.C.	285926	9364	
7	United Kingdom	London	58542	243	
8	Italy	Rome	57503	301	
9	Canada	Ottawa	31015	9971	
10	France	Paris	59453	552	
11	Germany	Berlin	82007	357	
12	Japan	Tokyo	127291	378	
13	Russia	Moscow	144664	17075	
14				Average	
15				Minimum	



 Scoring target cells

Figure 7: Example of scoring targets

4.2.1 Grading a Cell's Data

The most crucial problem in grading a cell is that many ways to get the correct answer exist. For instance, in Figure 7, let the scoring target cells be E6, E7, ..., E15. The ways to compute the correct value for E14 are in the following formulae:

$$\begin{aligned}
 &= \text{AVERAGE}(E6 : E13) && \text{(the sample answer)} && (2) \\
 &= \text{SUM}(E6 : E13)/\text{COUNT}(E6 : E13) && && (3) \\
 &= \text{SUM}(E6 : E13)/8 && && (4) \\
 &= (E6 + E7 + E8 + E9 + E10 + E11 + E12 + E13)/8 && && (5) \\
 &= 143.5294637314 && && (6) \\
 143.5294637314 &&& \text{(not strictly formula)} && (7)
 \end{aligned}$$

The most appropriate way to calculate a mean value is to use the function “AVERAGE” shown in equation (2). However, the module treats the values, except for equation (7) as the correct answer in the default settings. That is because of simplifying evaluations. PhpSpreadsheet can recognize a data type in a cell as a string, formula, numeric, or Boolean. The module with the default settings only checks the computation results in a cell and whether the data type is a formula.

4.2.2 A Control File for Grading

This subsection explains the module settings in the control file for MS-Excel. The module reads the control files using the function “parse_ini_file” in PHP [12]; then the format is based on the function. Figure 8 shows examples of the settings for cells, whose details are as follows:

- Single-line comment starts with “;.” We use “;---...” as a separator for better readability.
- The strings in square brackets “[]” represent cell names. A setting applies only to the cell.
- “__TYPE__” indicates the data type of the cell. It can be “s” (string), “n” (numeric), or “f” (formula).
- “__MODE__” means how to assess the cell. It has a non-negative integer value from 0 to 15, but the module internally uses it as a four-bit binary number. If “__MODE__” equals 0, the cell is not assessed. Each bit has the following effect if it is 1:
 - 1st bit:** Check the non-formatted value that is shown in “__VALUE__.”
 - 2nd bit:** Check whether the data type is correct according to “__TYPE__.”
 - 3rd bit:** Check whether the value is calculated using functions in “__BOUNDED_FUNC__.”
 - 4th bit:** Check the formatted value according to “__FORMATTED_VALUE__.”

The default setting of a cell that will be checked is $(11)_{10} = (1011)_2$.

- “__VALUE__” has a non-formatted value. If a cell contains functions, it is the result of the computation.


```

;-----
[E14]
__TYPE__=f
__MODE__=11
__WEIGHT__=1
__VALUE__="143.5294637314"
__OTHERS__="143.5294637314~~~143.5~~~0.5###143.5294637314~~~144~~~0.1"
__FORMATTED_VALUE__="143.53"
__FUNCTIONS__="=AVERAGE(E6:E13)"
__BOUNDED_FUNC__="AVERAGE"
__ERRORS__=""
;-----
[E15]
__TYPE__=f
__MODE__=11
__WEIGHT__=1
__VALUE__="3.110520509477485"
__OTHERS__=""
__FORMATTED_VALUE__="3.11"
__FUNCTIONS__="=MIN(E6:E13)"
__BOUNDED_FUNC__="MIN"
__ERRORS__="=MIN(E6:E14)"

```

Figure 8: Example of an MS-Excel control file

- “__OTHERS__” shows possible answers that can include partially correct answers. The answers are separated by “###,” and each answer has three parts that are delimited by “~~~.” The first part is the non-formatted value, the second part is the formatted value, and the third part has the ratio of the answer to the correct answer. In Figure 8, the correct answer of the cell “E14” is 143.53 as the displayed value (the actual value is 143.5294637314). If a student submits the answer 143.5, the answer receives a half grade. In the case a student submits the answer 144, the answer gets 0.1 times the correct grade.
- “__FORMATTED_VALUE__” has a formatted value. If the fourth bit of “__MODE__” is set, the module uses the value to check an answer.
- “__FUNCTIONS__” includes a formula of the sample answer. It has a value only when “__TYPE__” is “f.”
- “__BOUNDED_FUNC__” represents required functions as the correct answer. To use the parameter, the third bit of “__MODE__” must be set 1. In Figure 8, the cell “C14” requires the function “AVERAGE” for the correct answer.
- “__ERRORS__” has wrong answers. For example, the cell “E15” in Figure 8 has the correct answer “=MIN(E6:E13),” but if “E14” has “=AVERAGE(E6:E13),” “=MIN(E6:E14)” shows the correct value. The parameter is used to exclude such wrong answers.

Figure 9 shows examples of settings for a graph, whose details are as follows:

```

;-----
[CHART]
__NUM__=1
;-----
[CHART_1]
__CHART_MODE__=17
__KIND__="c:pieChart, , "
__STYLE__=""
__TITLE__="Population density per km2"
__CAT__="United States of America,United Kingdom,Italy,Canada,France,Germany,Japan,Russia"
__LEGEND__="Population density per km2"
__NUM_RANGE__=1
__RANGE_0__="'E-2022-0-0'!$E$6:$E$13"
__C_DATA__="1"
__DATA_0__="30.534600598035,240.91358024691,191.03986710963,3.1105205094775,..."

```

Figure 9: Control file for a graph

- The section “[CHART]” has “__NUM__” that shows the number of graphs that must be checked.
- The section name has the format “[CHART_?],” and “?” has a graph number such as 1, 2, ...
- “__CHART_MODE__” represents how to assess the graph. It has a non-negative integer value from 0 to 31, but the module internally uses it as a five-bit binary number. Each bit has the following effect if it is 1:
 - 1st bit:** check what kind of graph is shown in “__KIND__”.
 - 2nd bit:** check whether the title of graph is set as “__TITLE__”.
 - 3rd bit:** check x-axis in the graph according to “__CAT__”.
 - 4th bit:** check whether the data range to create the graph is the same as “__RANGE_?__” (“?” has a non-negative integer).
 - 5th bit:** check the graph style shown in “__STYLE__”.

The default setting of a cell that will be checked is $(9)_{10} = (01010)_2$.

4.2.3 Creation of Control Files

Creating control files is important for the module. However, inputting all settings is very hard, and mistakes can occur. Our system has a feature to create control files semi-automatically by uploading the correct answer file. Creating a control file has the following steps:

1. Create a sample answer file. Figure 10 shows a sample file to be uploaded.

	A	B	C	D	E
1	Exercise 00				
2	The following data is the population and the total area of Group Eight (G8).				
3	Fill the colored blank cells and write these numbers with two decimal places.				
4					
5	Country Name	Capital	Population (×1,000 people)	Total Area (× 1,000 km ²)	Population density per km ²
6	United States of America	Washington, D.C.	285926	9364	30.53
7	United Kingdom	London	58542	243	240.91
8	Italy	Rome	57503	301	191.04
9	Canada	Ottawa	31015	9971	3.11
10	France	Paris	59453	552	107.70
11	Germany	Berlin	82007	357	229.71
12	Japan	Tokyo	127291	378	336.75
13	Russia	Moscow	144664	17075	8.47
14				Average	143.53
15				Minimum	3.11
16	<p style="text-align: center;">Population density per km²</p> <div style="display: flex; justify-content: space-around; font-size: small;"> ■ United States of America ■ United Kingdom ■ Italy </div> <div style="display: flex; justify-content: space-around; font-size: small;"> ■ Canada ■ France ■ Germany </div> <div style="display: flex; justify-content: space-around; font-size: small;"> ■ Japan ■ Russia </div>				
17					
18					
19					
20					
21					
22					
23					
24					
25					
26					
27					
28					
29					
30					
31					
32					
33					

Figure 10: An MS-Excel sample answer file

- Upload the sample answer file. The cells that have formulae are automatically selected as candidates to be graded (Figure 11).

To add a cell to be graded, click a check box on the left side of the gray button.

	A	B	C	D	E
5	<input type="checkbox"/>	Country Name	<input type="checkbox"/>	Capital	<input type="checkbox"/>
6	<input type="checkbox"/>	United States of America	<input type="checkbox"/>	Washington, D.C.	<input type="checkbox"/>
7	<input type="checkbox"/>	United Kingdom	<input type="checkbox"/>	London	<input type="checkbox"/>
8	<input type="checkbox"/>	Italy	<input type="checkbox"/>	Rome	<input type="checkbox"/>
9	<input type="checkbox"/>	Canada	<input type="checkbox"/>	Ottawa	<input type="checkbox"/>
10	<input type="checkbox"/>	France	<input type="checkbox"/>	Paris	<input type="checkbox"/>
11	<input type="checkbox"/>	Germany	<input type="checkbox"/>	Berlin	<input type="checkbox"/>
12	<input type="checkbox"/>	Japan	<input type="checkbox"/>	Tokyo	<input type="checkbox"/>
13	<input type="checkbox"/>	Russia	<input type="checkbox"/>	Moscow	<input type="checkbox"/>
14	<input type="checkbox"/>			Average	<input type="checkbox"/>
15	<input type="checkbox"/>			Minimum	<input type="checkbox"/>
	<input checked="" type="checkbox"/>	Graph 1 detailed settings		Creation of scoring standard	questions

Figure 11: Semiautomatic creation of a control file

- To set details for a cell, click the cell data placed as a button. Figure 12 shows the settings for E14 to create the same as Figure 8. “__MODE__” is computed automatically by the settings.
- To set details for a graph, click the button “Graph 1 details setting”. If an MS-Excel file has multiple graphs, buttons appear according to the number of graphs. Figure 13 shows the settings for a graph. “__CHART_MODE__” is computed automatically by the settings.

Setting the scoring for cell E14

Scoring mode: 11

value: 143.5294637314

Value after formatting: 143.53

Score (weight):

Another solution:

value	Display value	ratio
<input type="text" value="143.5294637"/>	<input type="text" value="143.5"/>	<input type="text" value="0.5"/>
<input type="text" value="143.5294637"/>	<input type="text" value="144"/>	<input type="text" value="0.1"/>

a formula: = AVERAGE (E6: E13)

Mandatory function: AVERAGE

Wrong answer:

Figure 12: Detailed settings for a cell

Graph 1 scoring settings

Scoring mode: 17

Graph type: c:pieChart, .

Graph style:

Graph title: Population density per km2

Item axis label: United States of America,United Kingdom,It...

Graph legendw: Population density per km2

Range of data:

Data details: 1: 'E-2022-0-0'\$E\$6:\$E\$13 30.534600598035,240.91358024691,...

Figure 13: Detailed settings for a graph

4.3 Grading Module for MS-Word

The grading module for an MS-Word file can check the following items:

- formatting text: font type, font size, bold, italic, font color, underline, color or pattern behind a text, strikethrough, spacing between characters (expanding, condensing)
- paragraph settings: text align (left, centered, right, justified), indentation (left, right, first line, hanging), bulleted or numbered line, spacing (before, after), line spacing
- table settings: the number of rows and columns, text align (vertical, horizontal), merging cells, line types
- page settings: the number of lines, page margins, page orientation, text direction, the number of columns, page size, page style.

The module parses the data from a submitted MS-Word file according to Office Open XML file formats [13]. An MS-Word file is a zipped XML file, which the module parses using “The ElementTree XML API” in Python [14]. Figure 14 shows the parsed XML document as a tree.

4.3.1 Combining Split Text

Text of an MS-Word file is in “word/document.xml” as an XML. The XML consists of the <w:document> and <w:body> elements, followed by one or more block level elements

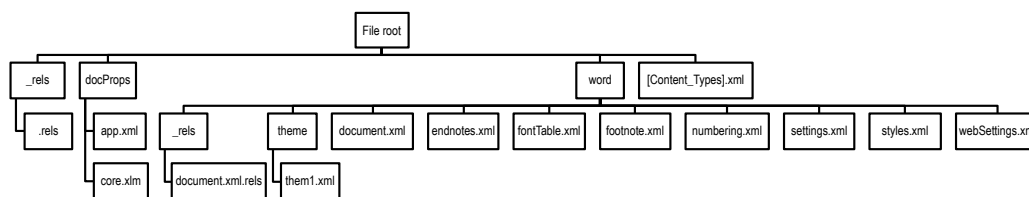


Figure 14: XML element tree of an MS-Word file

`<w:p>`, which represents a paragraph. A paragraph has one or more `<w:r>` elements. The `<w:r>` shows for run, which is a text region with a standard set of properties, such as formatting. A run contains not more than one `<w:t>` element. The `<w:t>` element contains a range of text [15]. However, the `<w:t>` elements are cluttered, and the texts are split into several elements. For example, an MS-Word file has a simple text “Hello World.” but its XML might have three “`<w:t>`” parts such as “`<w:r><w:t>H</w:t></w:r>`,” “`<w:r><w:t>ello </w:t></w:r>`,” and “`<w:r><w:t>World.</w:t></w:r>`”². That depends on the user’s input processes. Hence, according to the XML format, the module concatenates split text in several `<w:t>` tag in `<w:r>` tags which are included in the same `<w:rPr>` tag in order to check any formatting of the texts.

4.3.2 A Control File for Grading

This subsection explains the control file for grading MS-Word. The control file has the following elements:

- A comment starts with the hash character “#” and continues to the end of the line.
- Meta characters begin with “\” and end with “;:” They have special meanings, as explained below:

\XMLFILE=*path*;: This specifies a file path to an XML file in Figure 14. In most cases, it is “document.xml,” but “footnote.xml” is used to check a footnote.

\AND;; , \OR;; , \NOT;; , \END;; : These create a combinational question.

“\AND;;” requires that all the questions that follow the statement must be correct. “\OR;;” needs at least one of the questions to be correct. “\NOT;;” inverts a result of the question that follows the statement. “\END;;” indicates the end of the combinational question.

\STEM=*path*; , \STEX=*path*;: Both specify a path in an XML tree. The module finds the position of a question using the path. `\STEM` indicates a start point for searching XML elements and XML attributes. `\STEX` directs a start point of texts. These paths can be overwritten.

- A question consists of at least three lines:
 - The first line shows the number of a paragraph and a required element in an XML file.
 - The second line indicates a text position to be checked.

²In the other case, the text can be split into “Hello “ and “World.” or not be split.

- The third line has a comment.
- The fourth line includes a required format for the element.

Figure 15 shows an example of a control file. The correct sample answer file corresponding to the control file is in Figure 16. The control file requires a setting for the 10th paragraph and has three questions: setting the font type, font size and boldface.

```

\XMLFILE=word/document.xml;;
\STEX=/w:document/w:body/w:p[{}];;
\STEM=/w:document/w:body/w:p[{}];;

\AND;;
  10 rFonts
  10 1 67
  Font Type
  ascii=Times New Roman
;;
  \STEM=/w:document/w:body/w:p[{}]/w:pPr;;

  10 rFonts
  10 0 0
  Font Type
  ascii=Times New Roman
;;
\END;;

\STEM=/w:document/w:body/w:p[{}];;

  10 sz
  10 1 67
  Font Size
  val=24
;;
  10 b
  10 1 67
  Bold Text

```

Figure 15: Example of an MS-Word control file

The details are as follows:

- “10 rFonts” means that the 10th paragraph must have the element “rFonts”.
- “10 1 67” indicates the text that is 67 characters from the 1st character in the 10th paragraph.
- “ascii=Times New Roman” is the specified font type.
- The 2nd operand in the 1st question directs the setting for the whole paragraph.
- The question that starts with “10 sz” is almost the same as 1st question. It requires the setting of the font size.

- The 3rd question imposes the bold text for 67 characters from the 1st character in the 10th paragraph. The question does not need the 4th line.

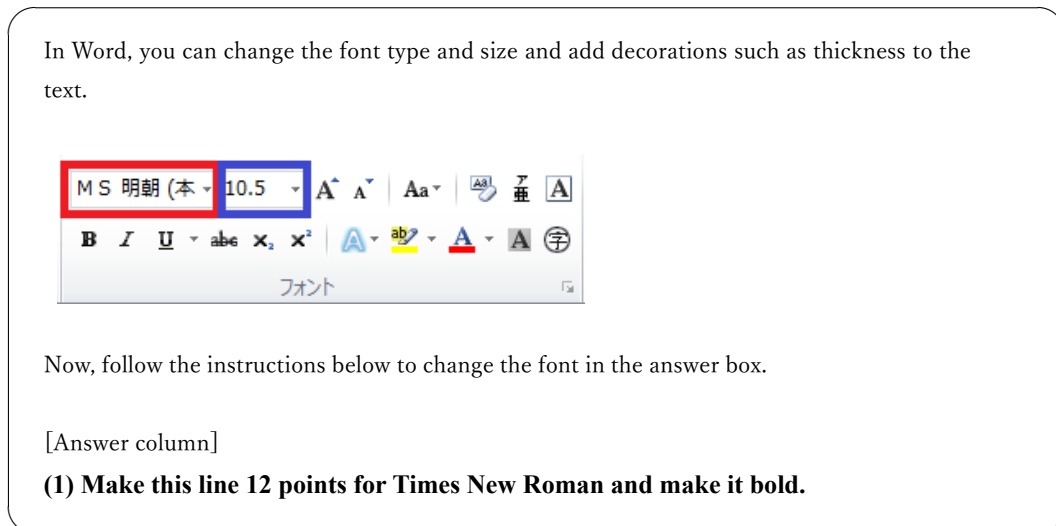


Figure 16: An MS-Word sample answer file

4.3.3 Creation of Control Files

Subsubsection 4.3.2 shows control files for grading MS-Word. Creating such control files requires proficiency in XML and Office Open XML file formats. Hence, our system provides a feature to help create files.

5 Experimental Results and Operational Results

Our system runs on the following environment:

CPU: Intel Xeon E3-1270 v6 3.80 GHz

Memory: 8GiB

Operating system: CentOS Linux 7.9-2009

Software: Apache 2.4.6, PostgreSQL 9.6.4, Python 3.6.2, PHP 7.3.3, PhpSpreadsheet 1.6.0

5.1 Performance of Modules

In evaluating the performance of the modules, we measured grading time ten times for MS-Excel 124 files, typewriting 107 files and MS-Word 126 files. The time required to grade one exercise is the different time calculated by the command “date +%s%N” used at the start and end times. Table 1 shows the results of grading time. Figures 17–19 are the distribution time for grading. It means our system works fast enough to grade students’ exercises.

Table 1: Grading time

	Average	Standard deviation (SD)	Median
Typewriting	16.89 μs	0.4254 μs	16.83 μs
MS-Excel	41.41 μs	2.163 μs	41.46 μs
MS-Word	14.72 μs	0.1381 μs	14.71 μs

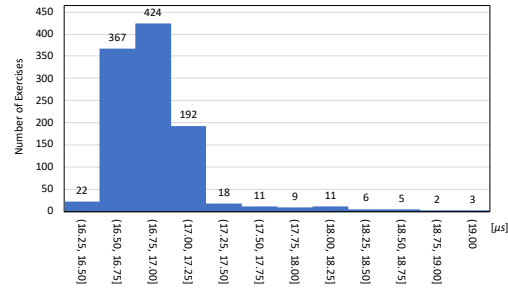
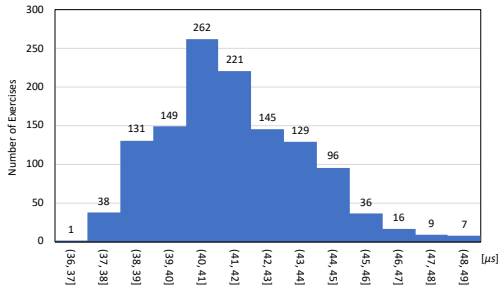


Figure 17: Distribution of MS-Excel grading time Figure 18: Distribution of typewriting grading time

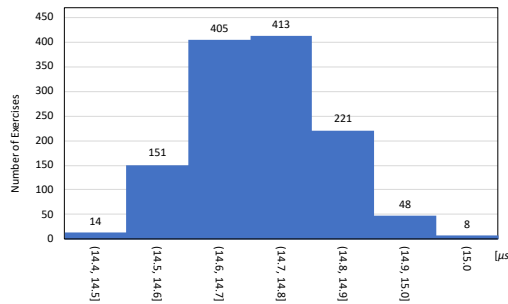


Figure 19: Distribution of MS-Word grading time

5.2 Immediate Feedback

Figures 20 and 21 show feedback from our system for an MS-Excel file and an MS-Word file, respectively.

The uploaded MS-Excel file to be graded has the following intentional mistakes:

- E6 is blank (has no answer).
- E7 has a correct value by formulae (“=C7/D7”), but is not displayed with two decimal places.
- E8 has 191.04, which is numeric and not a computed value by formulae.
- E14 has an incorrect value (the correct answer is 143.53).
- E15 has a wrong answer indicated by “__ERRORS__” (Figure 8).
- The graph is created with the range 'E-2022-0-0'!\$E\$7:\$E\$13.

	A	B	C	D	E
5	Country Name	Capital	Population (×1,000 people)	Total Area (× 1,000 km ²)	Population density per km ²
6	United States of America	Washington, D.C.	285926	9364	0.00
7	United Kingdom	London	58542	243	240.914
8	Italy	Rome	57503	301	191.04
9	Canada	Ottawa	31015	9971	3.11
10	France	Paris	59453	552	107.70
11	Germany	Berlin	82007	357	229.71
12	Japan	Tokyo	127291	378	336.75
13	Russia	Moscow	144664	17075	8.47
14				Average	159.67
15				Minimum	3.11

There are the following mistakes in the five places that are red. Please solve it again.

- Cell E6 is blank.
- The value in cell E7 is displayed incorrectly.
- No formula (function) is used to calculate cell E8.
- The value in cell E14 is incorrect.
- Cell E15 uses an invalid answer method.

There are the following mistakes regarding the graph. Please solve it again.

- The data range of the graph is specified incorrectly.

Figure 20: Feedback from grading an MS-Excel file

The MS-Word file requires that the font type is “Times New Roman”, the font size is 12 points and the text is made bold, but the upload file satisfies only the font type and the font size.

Correct answer 1st question 10th paragraph (1) Make this line 12 points for... **Font Type**
Correct answer 2nd question 10th paragraph (1) Make this line 12 points for ... **Font Size**
Incorrect answer 2nd question 10th paragraph (1) Make this line 12 points for ... **Bold Text** (wrong range of change)

Figure 21: Feedback from grading an MS-Word file

These figures mean that our system works properly and can give feedback immediately.

5.3 Operational Results

The introductory computer literacy courses in our university carry out two tests in a semester. The former is a pretest to check students’ skills before taking the course. The latter is a post-test to verify students’ proficiency. The difficulty level of these tests is the same. MS-Excel test checks whether a student can format values, use functions, and create a graph. MS-Word test checks whether a student can format texts and use paragraph settings.

We adopt the average normalized gain [16] to confirm all students’ skill growth. The average normalized gain G is calculated by equation (8):

$$G := \frac{\langle \%Post \rangle - \langle \%Pre \rangle}{100 - \langle \%Pre \rangle}, \quad (8)$$

where $\langle \%Pre \rangle$ and $\langle \%Post \rangle$ are the initial (pretest) and final (post-test) class average, respectively. The high average normalized gain G implies that the course provides a successful learning effect.

Table 2 shows the results of the pretest and the post-test of the students in 2021. Figures 22 and 23 are the 3D histograms of the tests. The results indicate that the course can

improve the students' computer skills. Especially, the decrease in MS-Excel's standard deviation means that students who received low scores in the pretest can earn higher scores later. Our system provides many exercises for students to improve their computer skills; therefore, it contributes more than a little to the success of the course.

Table 2: Average normalized gain

		Average	SD	Median	n	G
MS-Excel	Pretest	56.97	43.34	80.95	1,182	0.8482
	Post-test	93.47	21.95	100.00		
MS-Word	Pretest	28.78	17.60	26.19	1,242	0.5594
	Post-test	68.62	31.01	74.29		

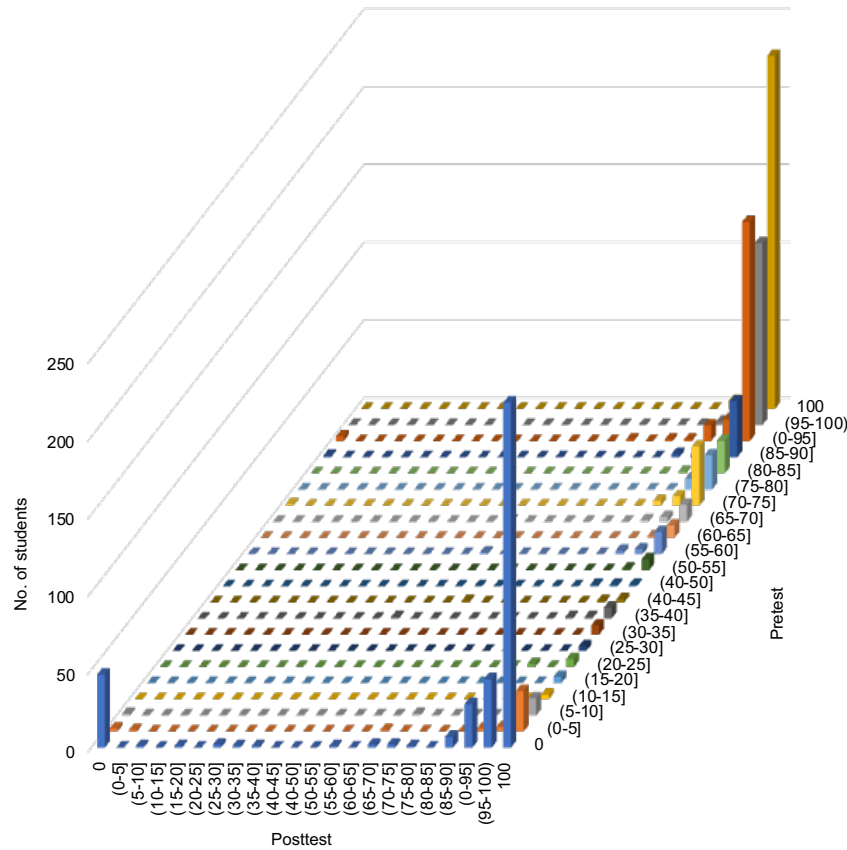


Figure 22: 3D histogram of MS-Excel pretest and post-test

6 Conclusion and Future Works

This paper explained the integrated learning system for first-year students to learn basic computer skills. The system can relieve teachers' workloads to grade many exercises of MS-Excel/MS-Word files. It can also provide immediate feedback and has a mechanism to prevent students from submitting copied files.

In addition, we showed the system's effectiveness from both perspectives: the time to grade MS-Excel, typewriting and MS-Word files, and the average normalized gain com-

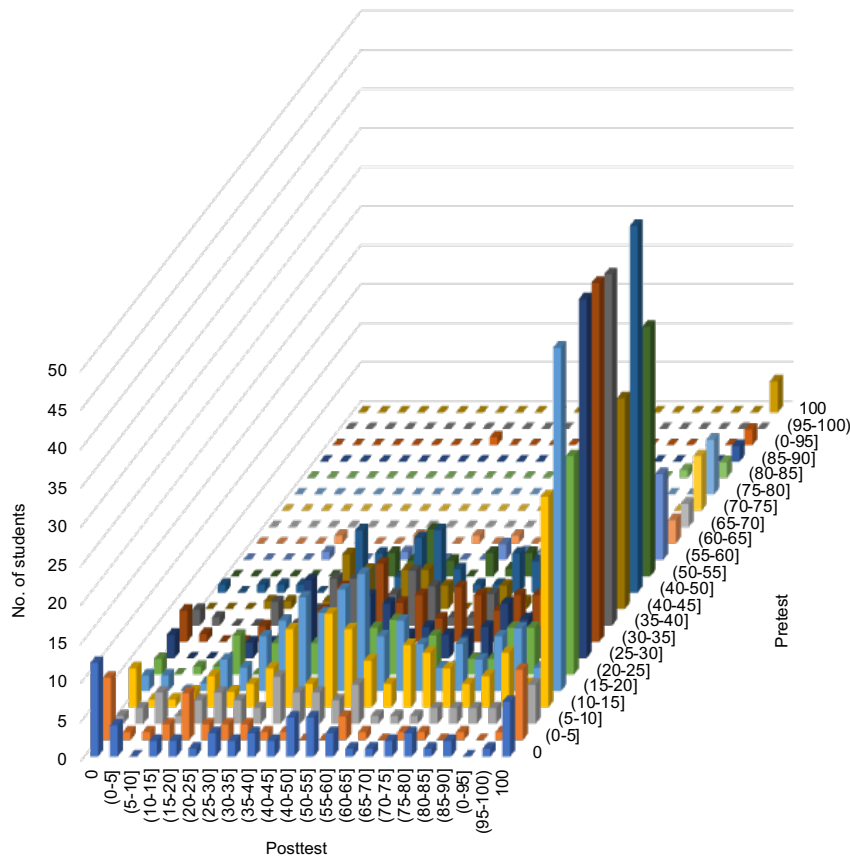


Figure 23: 3D histogram of MS-Word pretest and post-test

puted by the operation records of the system in our university in 2021. The results mean the variation between students’ computer skills decreased.

One of the future works is to improve the system portability under any operating systems. Our system depends on some libraries, so the installation of the system is hard work. We will introduce Docker to ease the installation. The other problem is creating control files for MS-Word. Our system provides only a simple tool to help the creation. Hence, we are going to improve the tool.

Acknowledgment

This research was supported by Nagoya General Education Laboratory in Aichi University as the “Creation and research of learning content for HITS”. The authors wish to acknowledge Prof. Emer. Katsuya Hasebe of Aichi University and Assoc. Prof. Masa-aki Taniguchi of Meijo University, for their help in implementing the system of this study. We are also grateful to the teachers in charge of the introductory computer literacy courses at Aichi University. The authors would like to thank Liyora Nakagawa for the English language editing.

References

- [1] Kazunori Iwata and Yoshimitsu Matsui. Implementation of an automated grading system for microsoft excel spreadsheets and word documents. In Tokuro Matsuo, editor, *Proceedings of 11th International Congress on Advanced Applied Informatics*, volume 81 of *EPiC Series in Computing*, pages 289–302. EasyChair, 2021.
- [2] Kevin Matthews, Thomas Janicki, Ling He, and Laurie Patterson. Implementation of an Automated Grading System with an Adaptive Learning Component to Affect Student Feedback and Response Time. *Journal of Information Systems*, 23:71–84, 06 2012.
- [3] Moodle. <https://moodle.org>.
- [4] Edmodo. <https://new.edmodo.com>.
- [5] Blackboard. <https://www.blackboard.com>.
- [6] Keith Hekman. Automated Grading of Microsoft Excel Spreadsheets. In *2019 ASEE Annual Conference & Exposition*, Tampa, Florida, June 2019. ASEE Conferences. <https://strategy.asee.org/32135>.
- [7] Douglas Kline and Thomas Janicki. Enhancing Economics and Finance Learning through Automated Grading of Spreadsheet Exercises. *JOURNAL OF ECONOMICS AND FINANCE EDUCATION*, 2(2):23–29, 01 2003.
- [8] Todd L. Cherry and Larry V. Ellis. Does Rank-Order Grading Improve Student Performance?: Evidence from a Classroom Experiment. *International Review of Economics Education*, 4(1):9–19, 2005.
- [9] Eugene W Myers. An O(ND) difference algorithm and its variations. *Algorithmica*, 1(1):251–266, 1986.
- [10] Sun Wu, Udi Manber, Eugene W. Myers, and Webb Miller. An O(NP) Sequence Comparison Algorithm. *Inf. Process. Lett.*, 35:317–323, 1990.
- [11] PhpSpreadsheet. <https://github.com/PHPOffice/PhpSpreadsheet>.
- [12] parse_ini_file – Parse a configuration file. <https://www.php.net/manual/en/function.parse-ini-file.php>.
- [13] Ecma International. ECMA-376, Office Open XML file formats, 5th edition. <https://www.ecma-international.org/publications-and-standards/standards/ecma-376/>.
- [14] The ElementTree XML. <https://docs.python.org/3/library/xml.etree.elementtree.html>.
- [15] Structure of a WordprocessingML document. <https://docs.microsoft.com/en-us/office/open-xml/structure-of-a-wordprocessingml-document>.
- [16] Richard R. Hake. Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses. *American Journal of Physics*, 66:64–74, 1998.