



Implementation of an Automated Grading System for Microsoft Excel Spreadsheets and Word Documents

Kazunori Iwata¹ and Yoshimitsu Matsui²

¹ Faculty of Business Administration, Aichi University, Nagoya, Aichi, Japan
kazunori@vega.aichi-u.ac.jp

² Faculty of Law, Aichi University, Nagoya, Aichi, Japan
matsui@vega.aichi-u.ac.jp

Abstract

This paper describes an automated grading system for MS-Excel files and MS-Word files for information technology education. 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 discuss the system's effectiveness from both perspectives: the time to grade MS-Excel/MS-Word files and the average normalized gain computed by the operation records of the system in our university.

1 Introduction

Microsoft Excel and Word (hereinafter referred to as 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 are recommended to take a course in MS-Excel and MS-Word. This course provides the students with basic skills in calculating data with functions, graphically presenting data, and formatting documents.

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 automated grading system for MS-Excel and MS-Word. The system can provide immediate feedback and has a mechanism to prevent students from submitting copied files. It has not only automated grading modules for MS-Excel

and MS-Word but also a typing-test module. For computer beginners, typing speed is essential, and the module for MS-Word does not include the typing-test function, so the system includes the typing-test module separately.

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.

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 [1].

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.

Hekman [2] explained the related works in detail. Our system is similar to Kline and Janicki for grading MS-Excel files [3] but treats the formulae that show the same results differently. The details are in subsection 3.5.1. In addition, our system also supports grading MS-Word files.

3 Implementation of an Automated Grading System

This section explains our automated grading 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 MS-Excel, MS-Word, and typing. 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.

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.

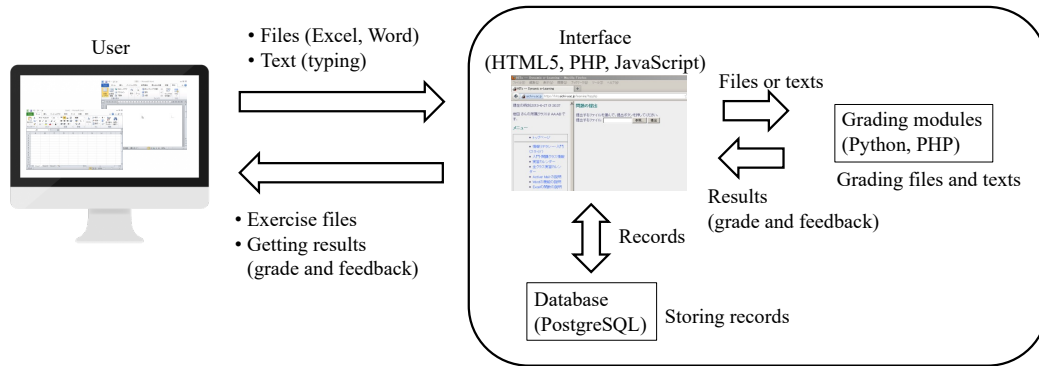


Figure 1: Outline of the system

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 typing module, 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.

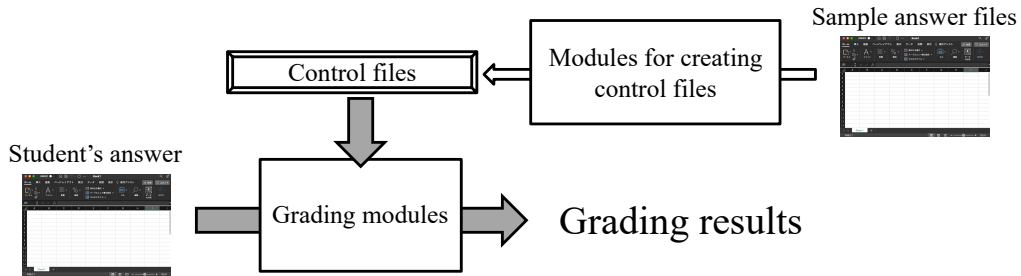


Figure 2: Scoring criteria

3.3 IDs of Exercises

The exercise IDs have the following format:

$$\alpha-x-y-z, \tag{1}$$

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 Typing, 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.

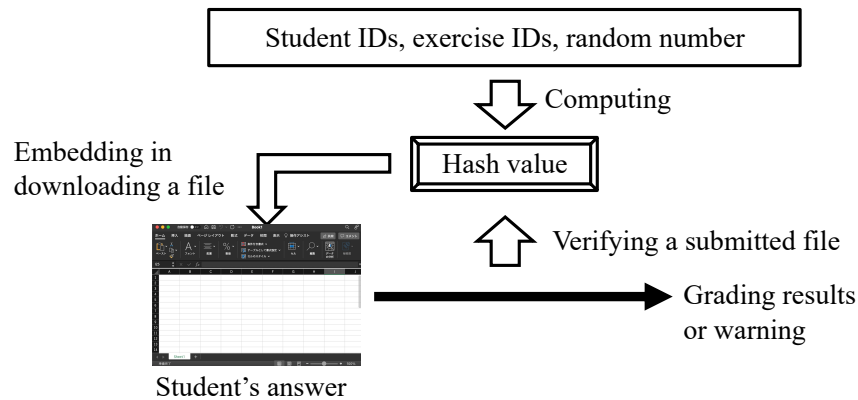


Figure 3: Embedding and verifying a hash value

3.5 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 [4].

In embedding a hash value, HITs locks cells that do not have scoring targets to protect them and colors scoring targets (Figure 4).

Exercise 00				
Fill in the colored cells				
Test scores	Japanese	Mathematics	English	Average
Tom	40	50	60	
Hanako	40	50	60	



 Scoring target cells

Figure 4: Example of scoring targets

3.5.1 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 4, let the scoring target cells be E5 and E6. The ways to compute the correct value for E5 are in the following formulae:

$$= \text{AVERAGE}(B5 : D5) \quad (\text{the sample answer}) \quad (2)$$

$$= \text{SUM}(B5 : D5) / \text{COUNT}(B5 : D5) \quad (3)$$

$$= \text{SUM}(B5 : D5) / 3 \quad (4)$$

$$= (B5 + C5 + D5) / 3 \quad (5)$$

$$= 50 \quad (6)$$

$$50 \quad (\text{not strictly formula}) \quad (7)$$

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.

3.5.2 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 [5]; then the format is based on the function. Figure 5 shows examples of the settings for cells, whose details are as follows:

<pre> ;----- [E16] __TYPE__=f __MODE__=11 __WEIGHT__=1 __VALUE__="143.5294637314" __OTHERS__="143.5294637314~~~143.5~~~0.5" __FORMATTED_VALUE__="143.53" __FUNCTIONS__="=AVERAGE(E8:E15)" __BOUNDED_FUNC__="AVERAGE" __ERRORS__="" </pre>	<pre> ;----- [E17] __TYPE__=f __MODE__=11 __WEIGHT__=1 __VALUE__="3.110520509477485" __OTHERS__="" __FORMATTED_VALUE__="3.11" __FUNCTIONS__="=MIN(E8:E15)" __BOUNDED_FUNC__="MIN" __ERRORS__="=MIN(E8:E16)" </pre>
---	--

Figure 5: Example of an MS-Excel control file

- 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 nonnegative 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:

First bit: Check the nonformatted value that is shown in “__VALUE__.”

Second bit: Check whether the data type is correct according to “__TYPE__.”

Third bit: Check whether the value is calculated using functions in “__BOUNDED_FUNC__.”

Fourth 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 nonformatted value. If a cell contains functions, it is the result of the computation.
- “__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 nonformatted value, the second part is the formatted value, and the third part has the ratio of the answer to the correct answer. In Figure 5, the correct answer of the cell “E16” 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.
- “__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 5, the cell “E16” requires the function “AVERAGE” for the correct answer.
- “`__ERRORS__`” has wrong answers. For example, the cell “E17” in Figure 5 has the correct answer “=MIN(E8:E15),” but if E16 has “=AVERAGE(E8:E15),” “=MIN(E8:E16)” shows the correct value. The parameter is used to exclude such wrong answers.

Figure 6 shows examples of settings for a graph, whose details are as follows:

- 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 nonnegative 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 the graph kind 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?__`”.

```

;-----
[CHART]
__NUM__=1
;-----
[CHART_1]
__CHART_MODE__=9
__KIND__="c:pieChart, , "
__STYLE__=""
__TITLE__=""
__CAT__="11-20,21-30,31-40,41-50"
__LEGEND__=""
__NUM_RANGE__=1
__RANGE_0__="'E-0-17-0'!$F$7:$F$10"

```

Figure 6: Control file for a graph

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

3.5.3 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 semiautomatically by uploading the correct answer file. Creating a control file has the following steps:

1. Create a sample answer file.
2. Upload the sample answer file. The cells that have formulae are automatically selected as candidates to be graded (Figure 7¹). To add a cell to be graded, click a check box on the left side of the gray button.

¹These figures are translated from Japanese to English by Google Chrome because our system currently supports only Japanese.

- To set details for a cell, click the cell data placed as a button. Figure 8 shows the settings for E16 to create the same as Figure 5. “_MODE_” is computed automatically by the settings.

Figure 7: Semiautomatic creation of a control file

Scoring settings for cell E16

Scoring mode: 11
 value: 143.5294637314
 Value after formatting: 143.53
 Score (weight):
 Alternative:

value	Display value	ratio
<input type="text" value="143.5294637314"/>	<input type="text" value="143.5"/>	<input type="text" value="0.5"/>
<input type="text" value="143.5294637314"/>	<input type="text" value="144"/>	<input type="text" value="0.1"/>

 a formula: = AVERAGE (E8: E15)
 Required functions: AVERAGE
 Wrong answer:
 Add column Delete column

Figure 8: Detailed settings for a cell

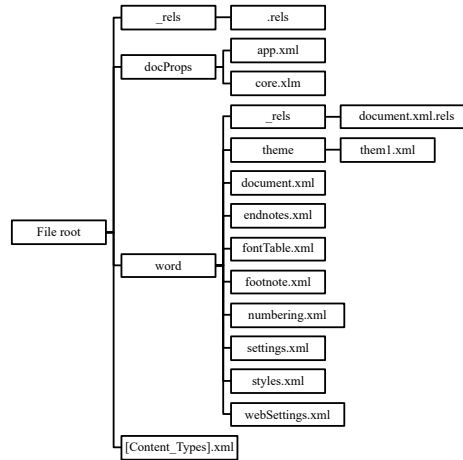


Figure 9: XML element tree of an MS-Word file

3.6 Grading Module for Typing

The grading module for typing has a simple feature to check students’ input. The module checks inputs after the student presses a “Grading” button after finishing an exercise. Most typing-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 [6, 7], 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.

3.7 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 [8]. An MS-Word file is a zipped XML file, which the module parses using “The ElementTree XML API” in Python [9]. Figure 9 shows the parsed XML document as a tree.

3.7.1 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 `<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 [10]. 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.

3.7.2 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 9. 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.

- A question consists of at least three lines:

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

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

Figure 10 shows an example of a control file. The file requires a setting for the tenth paragraph and has three questions. The details are as follows:

- “10 rFonts” means that the tenth paragraph must have the element “rFonts”.
- “10 1 97” indicates the text that is 96 characters from the first character in the tenth paragraph.
- “ascii=Times New Roman” is the specified font type.
- The question that starts with “10 sz” is almost the same as the first question. It requires the setting of the font size.
- The third question imposes the bold text for five characters from the 38th character in the tenth paragraph. The question does not need the fourth line.

3.7.3 Creation of Control Files

Subsubsection 3.7.2 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.

4 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

4.1 Performance of Modules

In evaluating the performance of the modules, we measured grading time ten times for MS-Excel 124 files, typing 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. 11–13 are the distribution time for grading. It means our system works fast enough to grade students’ exercises.

```
\XMLFILE=word/document.xml;;
\STEX=/w:document/w:body/w:p[10];;

10 rFonts
10 1 97
Font Type
ascii=Times New Roman
;;

10 sz
10 1 97
Font Size
val=24
;;

10 b
10 38 43
Bold Text
;;
```

Figure 10: Example of an MS-Word control file

Table 1: Grading time

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

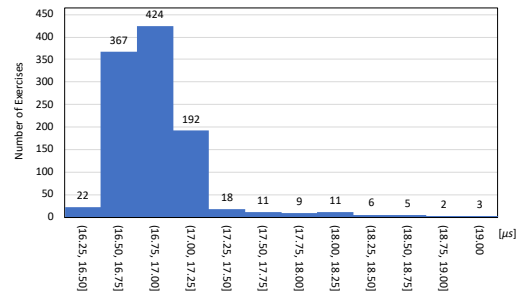
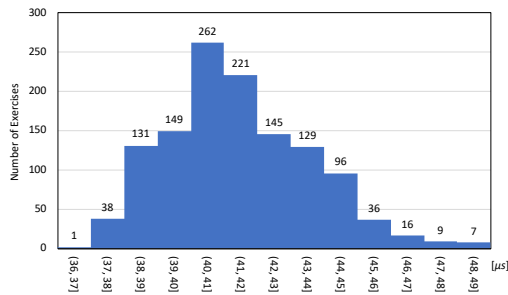


Figure 11: Distribution of MS-Excel grading time

Figure 12: Distribution of typing grading time

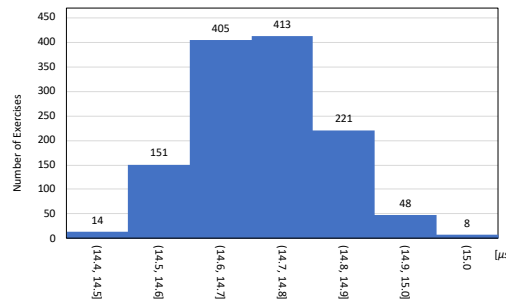


Figure 13: Distribution of MS-Word grading time

4.2 Immediate Feedback

Figures 14 and 15 show feedback from our system for an MS-Excel file and an MS-Word file, respectively ³.

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

- E8 is blank (has no answer).
- E9 has 240.91, which is numeric and not a computed value by formulae.
- E16 has an incorrect value (the correct answer is 143.53).
- E17 has a wrong answer indicated by “_ERRORS_”.

The MS-Word file requires that the font type is “MS Mincho” and the font size is 12 points, but the upload file satisfies only the font type.

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

³The feedback is also translated from Japanese to English by Google Chrome.

	A	B	C	D	E
7	Country name	capital	Population (thousands)	Area (thousand km2)	Population density
8	United States of America	Washington	285926	9364	0.00
9	England	London	58542	243	240.91
Ten	Italy	Rome	57503	301	191.04
11	Canada	Ottawa	31015	9971	3.11
12	France	Paris	59453	552	107.70
13	The Federal Republic of Germany	Berlin	82007	357	229.71
14	Japan	Tokyo	127291	378	336.75
15	Russian Federation	Moscow	144664	17075	8.47
16				average	143.50
17				minimum	3.11

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

- Cell E8 is blank.
- No formula (function) is used to calculate cell E9.
- The value in cell E16 is incorrect.
- Cell E17 uses an invalid answering method.

Figure 14: Feedback from grading an MS-Excel file

Correct answer 1st question 12th paragraph (1) Make this line 12 points for MS Mincho. **Font type**
Incorrect answer 2nd question 12th paragraph (1) Make this line 12 points for MS Mincho. **Font size**
(wrong range of change)

Figure 15: Feedback from grading an MS-Word file

4.3 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 posttest 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 [11] 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 (posttest) class average, respectively. The high average normalized gain G implies that the course provides a successful learning effect.

Table. 2⁴ shows the results of the pretest and the posttest of the students from 2015 to 2019. Figures. 16 and 17 are the distributions of the tests. The results indicate that the course can improve the students' computer skills. Especially, the decrease in both standard deviations means that students who received low scores in the pretest can earn higher scores later. Our

⁴ n in the table is the total number of students from 2015 to 2019.

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	<i>n</i>	<i>G</i>
MS-Excel	Pretest	65.90	41.20	90.48	6,355	0.9597
	Posttest	98.62	9.351	100.00		
MS-Word	Pretest	65.64	28.76	73.68	6,481	0.9122
	Posttest	97.33	12.02	100.00		

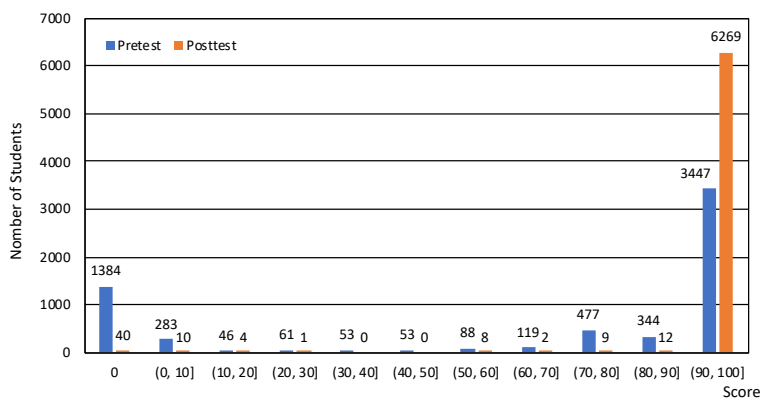


Figure 16: Distribution of MS-Excel pretest and posttest

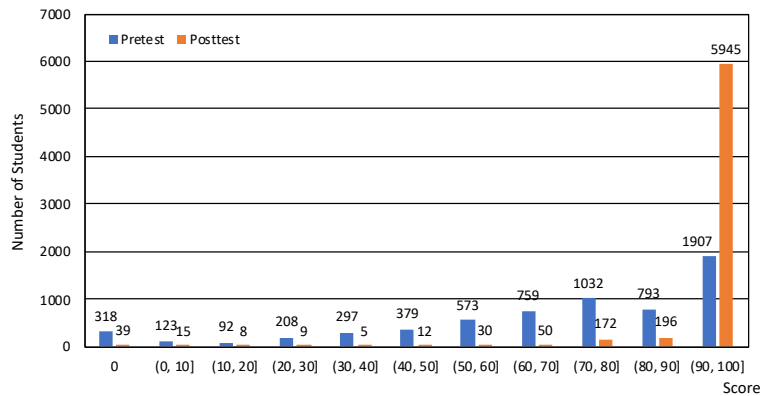


Figure 17: Distribution of MS-Word pretest and posttest

5 Conclusion and Future Works

This paper explained the automated grading system for MS-Excel files and MS-Word files for information technology education. 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/MS-Word files and the average normalized gain computed by the operation records from 2015 to 2019 of the system in our university. 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.

6 Acknowledgement

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 Meiji 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 MARUZEN-YUSHODO Co., Ltd. (<https://kw.maruzen.co.jp/kousei-honyaku/>) for the English language editing.

References

- [1] 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.
- [2] 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>.
- [3] 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.
- [4] PhpSpreadsheet. <https://github.com/PHPOffice/PhpSpreadsheet>.
- [5] parse_ini_file – Parse a configuration file. <https://www.php.net/manual/en/function.parse-ini-file.php>.
- [6] Eugene W Myers. An O(N^D) difference algorithm and its variations. *Algorithmica*, 1(1):251–266, 1986.
- [7] Sun Wu, Udi Manber, Eugene W. Myers, and Webb Miller. An O(N^P) Sequence Comparison Algorithm. *Inf. Process. Lett.*, 35:317–323, 1990.
- [8] Ecma International. ECMA-376, Office Open XML file formats, 5th edition. <https://www.ecma-international.org/publications-and-standards/standards/ecma-376/>.
- [9] The ElementTree XML. <https://docs.python.org/3/library/xml.etree.elementtree.html>.
- [10] Structure of a WordprocessingML document. <https://docs.microsoft.com/en-us/office/open-xml/structure-of-a-wordprocessingml-document>.
- [11] 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.