

On the Use of Regular Expressions and Exact-text Matching in Computer-based Test Scoring Algorithms

Wojciech Malec ^a

Institute of Linguistics, John Paul II Catholic University of Lublin, Al. Raclawickie, Lublin, Poland

Keywords: Regular Expressions, Exact-text Matching, Scoring Algorithms, Computer-based Language Testing.

Abstract: This paper addresses the issue of implementing two alternative approaches to developing automated scoring algorithms in computer-based language testing. One of the approaches discussed is based on the application of regular expressions and the other is based on exact-text matching. Although scoring algorithms that make use of regular expressions are technologically very attractive, there is evidence to indicate that they are not able to guarantee perfect scoring accuracy. Accordingly, in testing situations where decisions about the test takers need to be made on the basis of test scores, as in school settings, exact-text matching may actually be the preferred option. Moreover, whichever approach is adopted, it seems reasonably clear that automated scoring should ideally be subject to some human verification. The paper includes a brief description of a testing system where automated scoring employing exact-text matching is supplemented by human verification of the results.

1 INTRODUCTION


Computer-based language tests can be automatically scored using several approaches, which principally depend on the type of expected response. First, selected-response items, such as multiple choice or true/false, are commonly scored either by means of a rule known as ‘number right’ or through the use of ‘formula scoring’. The latter functions as a correction for random guessing (see Budescu & Bar-Hillel, 1993, for more on formula-scoring rules). Second, limited-production items, which include gap-filling and short answer, can be automatically scored by comparing the responses submitted by the test takers with the keyed answers. The comparisons may be made using either exact-text or regular-expression matching (Carr & Xi, 2010; Carr, 2014). Finally, automated scoring of extended-production tasks, such as essays, can be performed with the aid of artificial intelligence and machine learning techniques (e.g. Kumar & Boulanger, 2020). However, although automated essay scoring is generally faster, cheaper, and fairer than human scoring, and frequently claimed to be equally as reliable (Shermis, 2010, and references therein), the appropriateness of using it

alone in high-stakes situations has been called into question (see Weigle, 2013, for a review).

This paper focuses on the scoring of limited-production items, where the expected response ranges approximately from one word to one sentence. The paper discusses the use of regular-expression matching as an alternative to exact-text matching and points to the imitations of the former method. Finally, the article briefly presents a web-based testing system (WebClass) which combines automated scoring algorithms with human verification of their output.

2 REGULAR-EXPRESSION MATCHING

Regular expressions are sometimes utilized in computerized testing systems to score the test takers’ limited-production responses. For example, Moodle offers a 3rd-party plugin, which allows test developers to create the so-called Regular Expression Short-Answer questions (Rézeau, 2022). The key for a question of this type can be single regular expression, such as the one in (1a) or (1b) rather than a number of alternative (acceptable) responses (i.e.

^a <https://orcid.org/0000-0002-6944-8044>

bat, *cat*, and *rat*). This example clearly demonstrates that the use of regular expressions is simply a more economical solution compared to exact-text matching.

(1a) `[bcr]at`

(1b) `(b|c|r)at`

Admittedly, manual generation of regular expressions may be quite time consuming and error-prone. However, as a further step in the development of computerized short-answer scoring systems, Ramachandran, Cheng, and Foltz (2015) offer a method for automating the generation of regular expressions. For many developers of computer-based tests, including school teachers, solutions of this kind seem to be an absolute necessity.

It is also worthy of note that in the case of limited-production test items the usefulness of regular expressions for the development of scoring algorithms may depend on how precise the responses are expected to be. Specifically, in certain tests of language ability, a high degree of precision may be necessary: this is often the case with vocabulary and grammar tests. For example, although the phrase *the girl ride a bike* may be understandable, it violates the basic rule of subject-verb agreement in English. Similarly, although the phrases *sign on the dotted line* and *sign on a dotted line* are almost identical, the former one is a more native-like expression. In such situations, the number of acceptable responses may be fairly limited, which means that a complete scoring key can be based solely on exact-text matching, and it may not be necessary to resort to using regular expressions.

By contrast, in tests of reading or listening comprehension, where grammatical accuracy is not a priority, responses are typically accepted as correct as long as they contain certain key terms, or keywords, specified in the expected answer (the key). Therefore, in constructing a scoring key the test item writer may define a suitable regular expression that the scoring algorithm will use to check whether all of the necessary keywords are included in the responses provided by the test takers. Nevertheless, developing a perfectly reliable algorithm based on regular expressions is not without its problems. This issue is addressed in the next section.

2.1 Illustrative Example

Limited-production items may require test takers to provide short statements or simple definitions. For illustrative purposes, consider an item which elicits a short sentence expressing the idea that bikes are

vehicles with two wheels. The prototypical expected response might be as follows:

(2) A bike is a vehicle with two wheels.

In exact-text matching approaches, the statement given in (2) would be the keyed answer for this test item. However, the same idea can be expressed in many other ways, including sentences containing some grammatical errors (but perfectly understandable). Examples of such sentences are given below:

(3a) Bikes are vehicles which have two wheels.

A bike is a vehicle with two wheels.

The bike is vehicle with two wheels.

A bike is a vehicle that has two wheels.

Bike vehicles have two wheels.

Bike is vehicle on two wheels.

While all of the sentences in (3a) share a similar structure (e.g., they begin with a noun phrase containing the word *bike*, in either singular or plural form), this is by no means the only possibility, as evidenced by the following examples (again, not necessarily correct in terms of grammar), the last of which does not even contain a verb:

(3b) Vehicles called bikes have two wheels.

Two wheels are typical of vehicles like bikes.

Two wheels define vehicles such as bikes.

Number of wheel on a bike vehicle is two.

Vehicle with two wheels is called bike.

Bike – a two-wheel vehicle.

As an alternative to exact-text matching approaches, all of the sentences in (3a) and (3b) could be considered acceptable on the grounds that each of them contains the following keywords (three of which might be either singular or plural):

(4) bike(s), vehicle(s), two, wheel(s)

The keywords may appear in actual statements in the order given (as in 3a) or in a completely different one (as in 3b). Moreover, they may be optionally followed or preceded by other words.

In a computer-based testing system, the scoring algorithm could be designed to conduct a keyword matching operation, for example using the `preg_match()` function in PHP. For this to be possible, the pattern required by the function in question has to be a regular expression (or regex for short). The regex for the keywords given in (4) may be specified in the following way:

- (5) `/^(?=.*\bbikes?\b) (?=.*\b
vehicles?\b) (?=.*\btwo\b
(?=.*\bwheels?\b) .+/i`

A bike has one front and one rear wheel.
Bike – a two-wheeled vehicle.

The regular expression provided in (5) is within slashes, followed by the `i` flag at the end, which indicates that the search should be case-insensitive. The caret (`^`) at the beginning is an anchor that denotes the start of the string (although it may be optional). The main part of the regex consists of four capturing groups (inside parentheses), each corresponding to one of the keywords in (4). Every keyword requires a positive lookahead (denoted by `?=`) and is also preceded by the dot (`.`) metacharacter and the asterisk (`*`) quantifier (which together match any character between zero and unlimited times, thus making it possible for the response to optionally include some other words, in addition to the keywords). Moreover, each keyword is between word boundaries (indicated by `\b`). Three of the keywords have additionally a question mark at the end (which means that the preceding `s` is optional in each case). Finally, the last capturing group is followed by a dot and a plus sign (`+`). These two symbols match any character at least once (between one and unlimited times). In effect, the dot and plus match the entire expression on condition that all of the positive lookahead assertions are true.

The regex in (5) is capable of matching each of the sentences given in (3a) and (3b). In view of the fact that, using exact-text matching approaches, these sentences would all have to be included in the key (i.e. in an array of alternative answers that deserve full credit), the keyword approach based on regular expressions is a neat solution. Despite this, however, it is far from perfect as the regex in (5) would also match sentences which are too vague to be accepted as correct, for example:

- (6) Vehicles including bikes have two wheels.
Bikes denoting vehicles have two wheels.

To make matters worse, it would match sentences which are definitely incorrect, for example:

- (7) Bikes are not vehicles with two wheels.
Bikes are vehicles with two or five wheels.
Bikes are vehicles with more than two wheels.
Bikes are vehicles with twenty two wheels.
Every vehicle with two wheels is a bike.

The opposite situation is also possible: some sentences which correctly define bikes would not be matched by the regex. Examples are given below:

- (8) Bikes have a pair of wheels.
A bike is a vehicle with a pair of wheels.

It should be clear that the regex in (5) requires modifications as it is not capable of matching every possible correct response. Amongst other necessary changes, the keyword *vehicle* would need to be made optional. More importantly, however, even if the regex is successfully adjusted to match all of the examples presented above, it may be difficult to rule out the possibility of someone producing yet another alternative (and acceptable) response. One situation in which the use of regular expressions may be particularly challenging is when the keywords include a word for which the number of acceptable synonyms can be very large. The adjective *good* is a case in point.

2.2 Regular Expressions Versus Exact-text Matching

The question that arises in this context is whether the use of keywords and regular expressions is a better solution than exact-text matching. On the one hand, it must be admitted that keyword matching is very likely to result in fewer errors compared to exact-text matching. On the other hand, even if the keyword method results in only one mismatch (and the exact-match method generates dozens of mismatches), there remains the problem of identifying that single mismatch in a set of responses submitted by the test takers. And even if there are actually no mismatches, there should be a way of making sure that this is indeed the case. In all probability, the only solution is some kind of human verification of the automated scoring. Otherwise, some students may end up with inaccurate scores (and we might not even be aware of this).

However, if we accept that human verification is a necessity, then exact-text matching is actually superior to the keyword technique. The reason for this is that the keyword approach can potentially make two kinds of errors. As shown above, it can give full credit to incorrect responses, which might be called *false positive mismatches*, as in (7), or it can give no credit to correct responses, which could be termed *false negative mismatches*, as in (8) above. The exact-match approach, by contrast, can only make one type of error, namely false negative mismatches. This is because, in the exact-match approach, it is impossible for an answer to be included in the key and actually deserve no credit (unless the answer is there by mistake).

Suppose, for instance, that in the exact-match approach we have an item with four different keyed answers. Now, if test takers provide ten different

responses, four of which match the key, then only the remaining six will need to be taken care of in the process of verification to determine whether or not they are false negative mismatches. In the keyword approach, by contrast, all of the ten responses will need to be verified because, potentially, each of them can be either a false positive or a false negative mismatch.

In a sense, scoring systems that require human verification could be regarded as detracting from the attractiveness and utility of computer-based testing. Unfortunately, it does not seem to be possible for automated scoring to be one hundred percent accurate, at least in the case of certain constructed-response test items, and some kind of verification is usually indispensable. This should not be taken to suggest that humans make fewer errors than computers. In fact, the opposite may well be true as computer-based tests are generally more reliable in that they make the scoring more consistent over time and across test forms (e.g. Brown, 2016). However, the point is that computers may not be able to make the final decisions as to the lexical and grammatical correctness of constructed responses. Indeed, at least in school settings, it is teachers rather than computers who are ultimately responsible for the scoring. Accordingly, when teachers mark a response as incorrect, it is their task to explain to the students why it has not been accepted. If computerised testing systems were to completely replace humans, they would need to be perfectly accurate, and this may be hard to achieve in cases where responses have to be produced rather than selected.

3 WebClass

WebClass is a homegrown online platform that combines a MySQL database with PHP scripts running on a LiteSpeed web server (Malec, 2018). It can be classified as an academic learning management system (cf. Foreman, 2018) integrating features such as learner management, communication, content authoring, and assessment. It has recently been used primarily as a web-based assessment tool to develop online language tests for university and secondary school students (Malec & Malec, 2021). The testing component of the platform can be utilised to create tests and quizzes of several different types (including selected- and constructed-response items as well as extended-response tasks), administer them to students, monitor the test-taking process, mark the responses automatically (but with a possibility of verifying and overriding the scoring),

provide general and answer-specific feedback to the test takers, analyse tests and items statistically, store items in the item bank and then reuse them with the aid of a test generator or by importing them into existing tests (see also, e.g., Malec, 2015; Marczak, Krajka, & Malec, 2016).

3.1 Scoring Algorithm

Limited-production items are scored on WebClass by comparing each response submitted by the test takers with the keyed answers. For reasons discussed above, the comparison is made using exact-text matching rather than keyword or regular-expression approaches. An example is provided below of how scoring works in practice, followed by a discussion of human verification of automated scoring.

The operation of the scoring algorithm as applied to limited-production items is illustrated using this example from a language test:

(9) **Use the Word Given**

Complete the second sentence so that it means the same as the first sentence, using the word given.

This is, undoubtedly, the best road trip in my life. **SHADOW**

This is, _____, the best road trip in my life.

KEY

- (1) without a shadow of a doubt
- (2) beyond a shadow of a doubt

Additional Settings

Spelling errors permitted for partial credit: 2
Ignore: capitals, spaces, punctuation

Two keyed answers are provided for the item in (9), but in theory there is no limit to the number of acceptable alternatives. Additional settings, which are relevant to the scoring algorithm, include the number of spelling mistakes permitted for partial credit (half a point) as well as instructions to disregard capitalisation, spacing, and punctuation. The number of spelling errors allowed for half a point depends on the length and type of the expected answer. In some cases, even one spelling mistake is undesirable. For example, when the expected answer is a short word, such as the preposition *on*, changing it to, for example, *in*, would not be a minor error that deserves partial credit. It is thus important to make sure that the partial-credit error level specified for a given item does not result in unacceptable responses being awarded half points.

From a programming point of view, the scoring algorithm determines the number of spelling mistakes

using the `levenshtein()` function in PHP (more on this function can be found in, e.g., Quigley & Gargenta, 2007; see also Lisbach & Meyer, 2013, on its use in linguistics). The function returns the so-called Levenshtein distance, also known as the edit distance, between two strings, i.e. the minimum number of edit operations (insertions, deletions, or replacements) required to convert one string into the other. Partial credit is awarded on condition that the Levenshtein distance between the response and any of the keyed answers does not exceed the level specified by the item constructor.

Some examples of responses to the item in (9) are given in (10) below (the errors are underlined):

- (10) Without a shadow of a doubt [1]
 without a shadow o fa doubt [1]
 beyond a shadow₂ of a doubt [1]
 beyond a shaddow of a doubt [0.5]
 without a shaddow of a doubt [0.5]
 witout any shadow of doubt [0]

Each response above is followed by a score automatically computed by the algorithm. In the first three cases, the responses are awarded full credit thanks to the fact that the algorithm is set to ignore capitals, spaces, and punctuation, respectively. The next two responses are awarded partial credit because the number of spelling errors is not greater than the number allowed. Finally, the last response receives zero points because it contains three spelling mistakes and is thus not similar enough to the key. Looking at the responses scored by partial credit in (10), it might be argued that the spelling errors are mere typos and that these answers actually deserve full credit. Moreover, the last response is not necessarily completely wrong, and it might be given partial (or even full) credit. These examples strongly suggest that automated scoring can be significantly enhanced by some kind of human verification.

3.1.1 Verification of Automated Scoring

In the testing system discussed here, score verification follows test administration. The relevant tool (illustrated in Figure 1) is a scrollable web page presenting the teacher or test administrator with the entire test, one item after another. Each item is followed by incorrect responses (i.e. those for which no matches have been found in the key) as well as the names of the test takers who submitted the responses (these names can optionally be hidden). For each such response, the automated scoring can be changed into partial or full credit. When the computer’s decision is overridden, the response is saved in an array of

answers which deserve either full credit or partial credit. The scoring algorithm first attempts to find a match in these two arrays prior to determining the number of spelling mistakes. If a match is found, the computation of the Levenshtein distance is skipped. Taking the above into account, the steps in the operation of the scoring algorithm can be defined as in Table 1. It should be added that each step is executed only if none of the previous conditions is met.

Table 1: The scoring algorithm on WebClass.

Step	Condition	Score
1	Response is empty	0
2	Response matches the key (optionally disregarding capitalisation, spacing, punctuation)	1
3	Response matches full-credit answers	1
4	Response matches partial-credit answers	0.5
5	Levenshtein distance does not exceed the level allowed for partial credit	0.5
6	None of the above is true	0

Several comments are worth adding about the score verification tool. First, responses which are entirely correct (and do not contain any typos) can be added to the key. Second, the arrays of full-credit and partial-credit answers can be created earlier, i.e. at the stage of test construction. However, during score verification, the decisions are based on real responses actually submitted rather than those which may potentially be given by the test takers. Third, changes made to the scoring of any given response have an effect on every test submitted by all the test takers – it is not necessary to repeat the procedure for each test taker. Fourth, score verification can only be applied to limited-production items – the scoring of selected-response items is fully automated and cannot be changed, whereas extended responses (such as essays) are manually marked using an HTML editor. Fifth, the verification tool can also be used to give feedback to the test takers in the form of general and answer-specific comments. Finally, the content of the verification tool can be viewed ‘by students’ (not ‘by items’), in which case a list is displayed of all the students (grouped by classes) who have attempted the test. For each individual, the test can then be opened in a popup window where the scoring of her or his responses can be verified and validated.

In classroom settings, some further uses for the score verification tool are possible. For example, teachers may use the web page (displayed on a screen or wall by means of a projector) to discuss the results with the whole class. When students do not see the tool, the names provided for each incorrect response

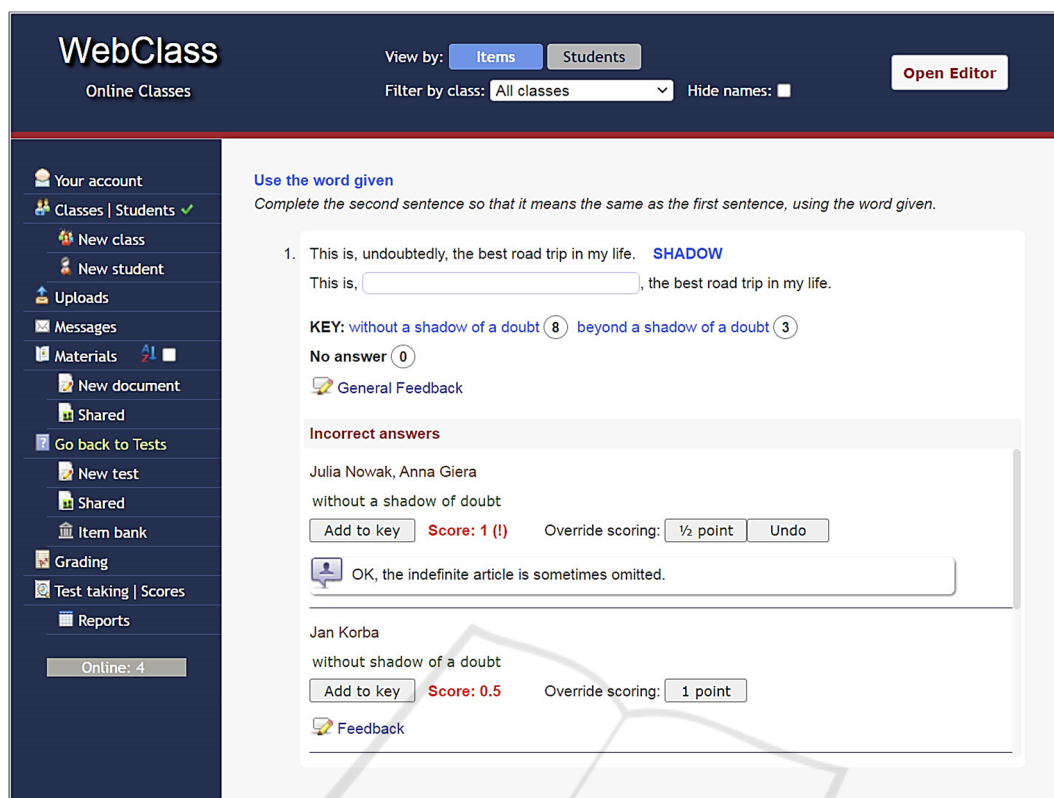


Figure 1: Verification of automated scoring on WebClass.

allow the teacher to easily address individual students.

4 CONCLUSION

One of the oft-quoted benefits of the use of technology for testing is that computer-based tests, as opposed to their paper-based counterparts, can be scored automatically. However, technology alone is not capable of guaranteeing high precision of scoring because automated scoring “can only be as accurate as the human-produced answer keys” (Brown, 2016). This article has discussed two approaches to producing answer keys to limited-production test items, namely exact-text matching and keyword matching (based on regular expressions). The main advantage of scoring algorithms implementing keyword matching is that a single regular expression can replace a whole array of keyed responses required for exact-text matching. On the other hand, it is easy to find examples of responses which ‘fall through the cracks’ inherent in solutions employing regular-expression matching. One possible conclusion to be drawn from this is that natural language is too

complicated for regular expressions to handle it impeccably.

However, if this conclusion is valid, the outcome of machine scoring evidently requires some human verification. Furthermore, as has been argued above, another reasonable conclusion is that exact-text matching (which can result in one type of errors) allows for more time-efficient score verification than does keyword matching (which can potentially result in two types of errors).

The second part of the article has presented the score verification tool available on WebClass, an online learning management system. This tool presents the teacher or test developer with incorrect responses only. It does not show correct responses because these are given full credit and do not need any further verification. As explained above, this solution is in accord with the exact-match approach. An alternative solution based on the keyword method would require the score verification tool to display all of the responses submitted by the test takers simply because every single response could potentially be inappropriately scored (no matter how unlikely this might be).

It is worth adding that the amount of time required for score verification on WebClass partly depends on

the level of difficulty of the test. If most of the test takers know the correct answers, there are relatively few incorrect responses that need any verification. By contrast, if most of the students produce incorrect responses, deciding which of them are appropriately identified by the scoring algorithm as wrong may require a considerable amount of time. The number of responses that may need verification also depends on the length of the expected answer. If the expected answer is a single word or a short phrase, the possibilities for alternative responses may be very limited. It is then also much easier to create a complete scoring key.

REFERENCES

- Brown, J. D. (2016). Language testing and technology. In F. Farr & L. Murray (Eds.), *The Routledge Handbook of Language Learning and Technology* (pp. 141–159). London: Routledge.
- Budescu, D., & Bar-Hillel, M. (1993). To guess or not to guess: A decision-theoretic view of formula scoring. *Journal of Educational Measurement*, 30(4), 277–291.
- Carr, N. T. (2014). Computer-automated scoring of written responses. In A. J. Kunnan (Ed.), *The Companion to Language Assessment* (Vol. 2, Chap. 64, pp. 1063–1078). London: John Wiley & Sons.
- Carr, N. T., & Xi, X. (2010). Automated scoring of short-answer reading items: Implications for constructs. *Language Assessment Quarterly*, 7(3), 205–218. doi:10.1080/15434300903443958
- Foreman, S. D. (2018). *The LMS Guidebook: Learning Management Systems Demystified*. Alexandria, VA: ATD Press.
- Kumar, V., & Boulanger, D. (2020). Explainable automated essay scoring: Deep learning really has pedagogical value. *Frontiers in Education*, 5. doi:10.3389/educ.2020.572367
- Lisbach, B., & Meyer, V. (2013). *Linguistic Identity Matching*. Wiesbaden: Springer Vieweg.
- Malec, W. (2015). The development of an online course in Irish: Adapting academic materials to the needs of secondary-school students. In A. Turula & M. Chojnacka (Eds.), *CALL for Bridges between School and Academia* (pp. 111–127). Frankfurt am Main: Peter Lang.
- Malec, W. (2018). *Developing Web-Based Language Tests*. Lublin: Wydawnictwo KUL.
- Malec, W., & Malec, M. (2021). WebClass jako narzędzie do kontroli i oceny postępów w nauce. *Języki Obce w Szkole*, 3/2021, 37–42.
- Marczak, M., Krajka, J., & Malec, W. (2016). Web-based assessment and language teachers – from Moodle to WebClass. *International Journal of Continuing Engineering Education and Life-Long Learning*, 26(1), 44–59.
- Quigley, E., & Gargenta, M. (2007). *PHP and MySQL by Example*. Upper Saddle River, NJ: Pearson Education, Inc.
- Ramachandran, L., Cheng, J., & Foltz, P. (2015, June 4). *Identifying patterns for short answer scoring using graph-based lexico-semantic text matching*. Paper presented at the Tenth Workshop on Innovative Use of NLP for Building Educational Applications, Denver, Colorado.
- Rézeau, J. (2022). Regular Expression Short-Answer question type. Retrieved from https://docs.moodle.org/311/en/Regular_Expression_Short-Answer_question_type
- Shermis, M. D. (2010). Automated essay scoring in a high stakes testing environment. In V. J. Shute & B. J. Becker (Eds.), *Innovative Assessment for the 21st Century* (pp. 167–185). New York: Springer.
- Weigle, S. C. (2013). English language learners and automated scoring of essays: Critical considerations. *Assessing Writing*, 18(1), 85–99. doi:10.1016/j.asw.2012.10.006