AUTOMATING THE CONSTRUCTION OF SELECTED-RESPONSE ITEMS WITH A TEXT-TO-ITEMS CONVERTER

Wojciech Malec[1]

Abstract: This paper focuses on the role of technology in facilitating the process of language test development in online settings. In particular, it takes an in-depth look at one specific step in the entire testing cycle: the construction of (selected-response) test items with the aid of the text-to-items converter on WebClass, the author's own online learning management system (webclass.co). The text converter can be used to edit an entire set of questions in a single editor pane (similar to a word processor) and then submit them to a parsing script which converts them into test items proper. The main advantage of using the converter is time efficiency: instead of moving step-by-step from one item to the next (which may be time-consuming), a large number of test items can be created in one go.

UDC Classification: 37 **DOI:** http://dx.doi.org/10.12955/cbup.v4.866

Keywords: language testing, web-based test construction, selected-response items, text-to-items converter, WebClass.

Introduction

The advantages of computerized tests over paper-and-pencil tests (PPTs), such as cost saving, automated data collection, simplified scoring, immediate reporting, greater measurement efficiency, innovative item types, and technological provisions for visually-impaired test takers (Parshall, Spray, Kalohn, & Davey, 2002), also apply to web-based tests (WBTs), traditionally defined as computer-based tests (CBTs) delivered over the internet (Roever, 2001). In addition to these benefits, WBTs may potentially be less time-consuming to construct than traditional PPTs, especially when test items are retrieved from an item bank. WebClass (Malec, 2012) offers the possibility of creating item banks for storing items that have already been administered and analyzed. In the first step, however, individual items have to be constructed, either by being written from scratch or by being converted from (pre-formatted) text. The focus of this article is on item writing, which "continues to be the most expensive and most time-consuming aspect of test development" (Haladyna, 2013, p. 13). In some aspects, such as cloze generation, the system described here can be used to create test items automatically (see Gierl & Haladyna, 2013, for more on automatic item generation).

Online Testing

Online tests can be constructed and delivered in a number of ways. In the simplest form, test items can be put in an HTML file, the key included in a JavaScript file, styles and images added, and all of the files uploaded to a web host. Students can respond to questions and quickly check their answers by clicking a button. Alternatively, the key can be stored in a database, students may respond to questions and submit the answers, which can be checked "on the fly" or later downloaded by the teacher for marking and analysis. The main drawback of such solutions[2] is that they require, at least basic, web development skills, and the entire testing cycle can be very time-consuming. A far better option is to use an online tool specifically designed for test development. Well-known examples of these, outlined by Douglas (2010), include "Hot Potatoes", "Moodle", and "WebCT" (now owned by "Blackboard").

E-testing on WebClass

WebClass (webclass.co) is an online learning management system (LMS) incorporating a test development component, which includes tools for test authoring, administration, and analysis. The system, whose core is made up of a set of PHP scripts and a MySQL database, allows teachers to register online classes, in which learners enroll by completing a registration form; construct, administer, mark, and analyze tests; and provide feedback, both collective and individual (answer-

[1] Wojciech Malec, Institute of English Studies, John Paul II Catholic University of Lublin, Lublin, Poland, malew@kul.pl
[2] For examples see http://webclass.co/info/e-testing.php.

specific). Students who join an online class can take e-tests and quizzes, view the scoring and feedback, and access a summary of all assessments with the final aggregate score.

The Development of Test Items on WebClass[3]

The cycle of language test development is traditionally divided into three stages: design, operationalization, and administration (Bachman & Palmer, 1996; see also Fulcher, 2010). Writing or modifying the actual test items, as well as test compilation, belong to the second stage. In general, the typical item development process takes the form of the following cycle on WebClass: constructing items, assembling items into test forms, administering the test(s), analyzing the items (on the basis of scores), storing selected items in a bank, and importing items from the bank. When existing items are retrieved from the bank, they may be modified, as required, and the entire cycle is repeated.

WebClass allows testers to construct items in over a dozen different formats, which can be grouped into two broad categories: selected- and constructed-response items. The former include Multiple-Choice (MC), True-False (TF), Multiple-Choice Cloze (MC-CL), Multiple-Response (MR), and Matching (MT) items.[4] The main advantage of all selected-response items is that they can be objectively scored. These popular items can be used at all levels of proficiency to test a wide range of content types, for example English collocations (Malec, 2015), and the principles of their construction have received much attention in the literature (see Haladyna, Downing, & Rodriguez, 2002, for a review).

Each MC item consists of the stem, e.g. “The children ___ in the park”, and the options (choices), e.g. “was/were/been”. The options include the key (correct answer) and distractors (incorrect choices). Although there is evidence that the optimal number of MC options is three (e.g. Bruno & Dirkzwager, 1995; Landrum, Cashin, & Theis, 1993; Trevisan, Sax, & Michael, 1994), for the purpose of pilot testing, it might be useful to have more, simply to identify those distractors which perform best. On WebClass, the default number is three, but this may be modified so that up to six options can be included in each item.

Constructing Selected-Response Items Using the Text Converter

The text converter on WebClass consists of two elements: a text editor and a parsing script. The text editor pane opens as a “popup” window from the main test editor; it can be used to edit a piece of text and prepare it for the parser to convert into test items proper. Technically, the editor area of the text converter is a DIV element with the contenteditable attribute, which means that the content can be live-edited directly in the browser. The basic steps in using the text editor pane can be summarized in the following way:

1. Type in or paste some text into the editable area.
2. Put square brackets around the words that you would like to replace with gaps (e.g. “The book is on [the] table”). (TIP: You can simply double-click a word to convert it into a gap.) The word in brackets is the key. There can be more words (or phrases) in the gap, separated by a forward slash:
   - MC (up to six): the first one is the key, the others are the distractors. When the test item is created, all options are randomized (unless the “Shuffle MC options” checkbox is unselected). Example: “The book [is/be/are] on the table”.
   - MR (no limit): if an option is incorrect, put an asterisk in front of it. Example: “The [book/cat/dog/lion/elephant/*clear] is on the table”.
   - Gap-filling[5] (no limit): there are no distractors here, so all of the words must be correct. Example: “The [book/radio/dog/bag] is on the table”.
3. When done, click “Create MC” (for MC items) or “Create FG/MT” (for gap-filling and MT items).

[3] The sections that follow are a modified version of the tutorial at http://webclass.co/info/formats.php.
[4] In fact, there is one more selected-response item format on WebClass, namely Right-Wrong (RW). However, since it is very similar to the TF item format, it is not included here.
[5] These are constructed-response items, but included here for the sake of completeness.

Multiple-Choice

Figure 1 shows three sample sentences in the editor area, each with prepositions of place inside square brackets.

These three sentences (Figure 1) are meant to be converted into a set of MC items, with the square brackets as gaps containing the options. When the "Create MC" button is clicked, the parsing script first looks for the gaps, around which test items are constructed; the text before and after the gap (the stem) is saved in the database as the context for the options (which are replaced with several underscore characters). The output of the parser is given in Figure 2.

Figure 1: The editor pane of the text converter on WebClass

Source: Webclass (a) (n. d.)

The main test editor illustrated in Figure 2 includes additional features, such as a JavaScript HTML editor, which can be used to modify the formatting of the items (e.g., text color and font size) and insert multimedia elements (images, audio, and video). It also provides the option of aligning MC choices vertically, as well as previewing the test items in their final form (see Figure 3).

It is worth noting that the system also works in reverse: it can convert existing test items (such as those in Figure 2) into (formatted) text, automatically pasted into the text editor pane (Figure 1) for revision and modification.

True-False

TF items are created in exactly the same way as MC. The only difference is that they have only two options, and these are the same for each item. An example is given in Table 1.

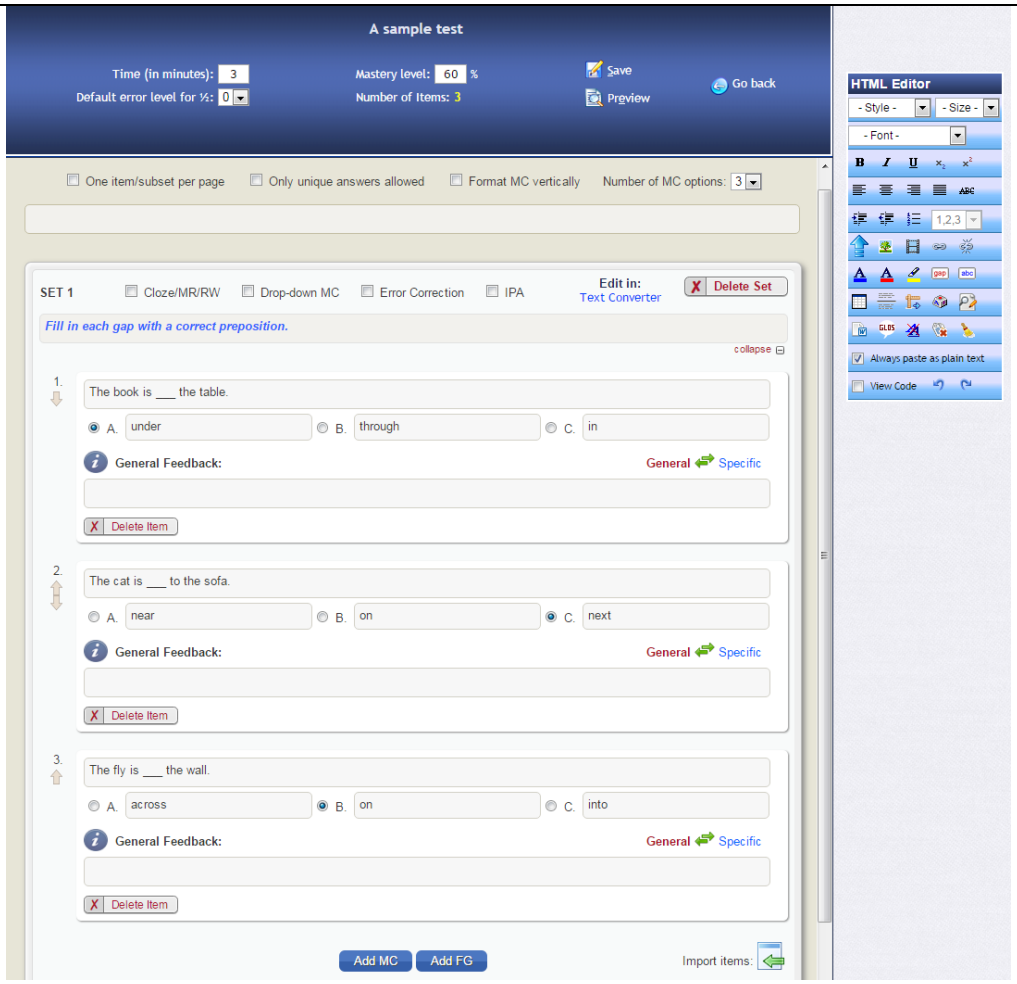
Table 1: The formatting of TF items

London is the capital of the UK. [true/false] Mars is larger than Venus. [t/f] Javascript is a markup language. [False/True] Dogs are more intelligent than cats. [F/T]

Source: Author

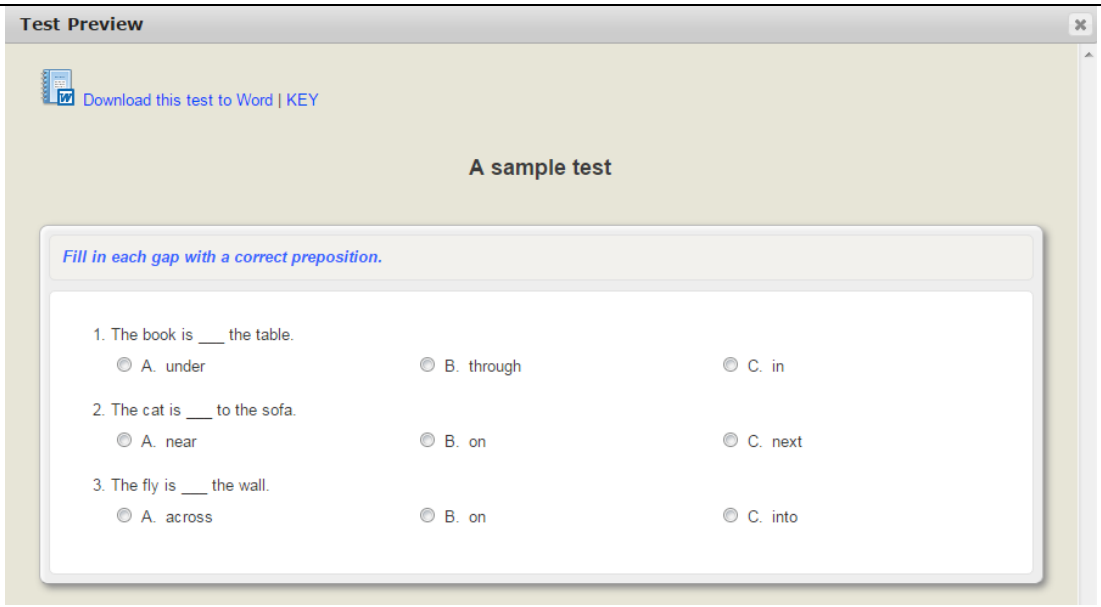
As in the case of MC items, the first option is the key. The parsing script recognizes 'true/false' and 't/f' (in either lower or upper case) as TF options.

Figure 2: Individual items in the test editor on WebClass



Source: Webclass (b) (n. d.)

Figure 3: Test preview on WebClass



Source: Webclass (b) (n. d.)

Multiple-Choice Cloze

Following the rules of creating standard MC items, the context sentences may be formatted inline (on the same line) so that they look like a passage of text rather than a numbered list. If a line break is needed, it can be inserted by pressing ENTER where necessary. An example is given in Table 2.

Table 2: An example of MC-CL
A postal carrier is working on a new [beat/blow/hit]. He comes to a garden gate marked BEWARE [OF/FROM/AT] THE PARROT! He looks down the garden and, [sure/certain/obvious] enough, there's a parrot sitting on its perch. He has a little chuckle to himself [at/on/for] the sign and the parrot there on its perch. The mailman opens the gate and walks into the garden. He gets as [far/long/closely] as the parrot's perch, when suddenly, it calls out: "REX, ATTACK!"
Source: Plannedparrothood (n. d.)

The parsing script splits the passage into as many individual items as there are gaps in the text, but these are joined together again when the test is displayed for the test takers. Additionally, to make the test as similar as possible to the original text, the options are given in the form of drop-down lists (see Figure 4).

Figure 4: MC-CL (preview)

Source: Webclass (c) (n. d.)

The construction of MC-CL can be further automated by means of the "gap-creator" tool in the text converter. For example, a passage of text can be used for testing the use of definite and indefinite articles in English. With the help of the gap-creator tool, every article in the text can be automatically surrounded by brackets, and then two more options can be added to the keyed responses (see Figure 5).

The example in Figure 5 shows that, additionally, the options "[--/a/the]" have been added in selected places where no articles should be used.

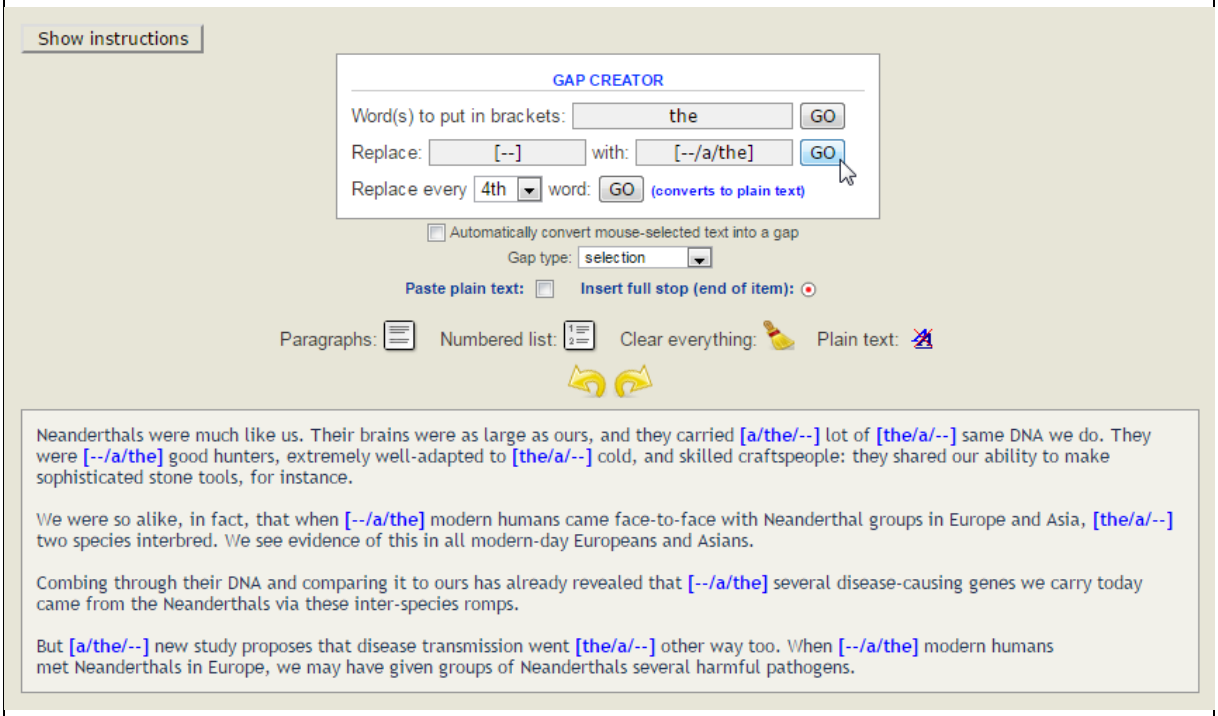
Multiple Response

MR items are also known as Multiple-Correct items. They are similar to MC items in that they also consist of a stem and several options. The difference between these formats is that in the case of MC there is one (and only one) correct answer, whereas in the case of MR no such restrictions apply.

What exactly counts as an item in an MR task may be somewhat problematic: is it the stem with all of the options or should each option be regarded as a single item? On WebClass, the latter solution has been adopted, which means that one point is awarded for each correct decision on every single option. If a test consists of five stems, each followed by four options, the maximum number of points to score is 20.

In the text converter, the incorrect choices are preceded by an asterisk, otherwise MR items are formatted similarly to MC items (see Table 3).

Figure 5: Using the gap-creator tool



Source: Webclass (d) (n. d.)

Table 3: The formatting of MR items

1. He kept [*handing/giving/paying/*telling] me compliments on my cooking.
2. I [swallowed/*devoured/*ate/*consumed] my pride and did as I was told.
3. I recognize his face, but his name [*fails/escapes/*misses/*forgets] me.

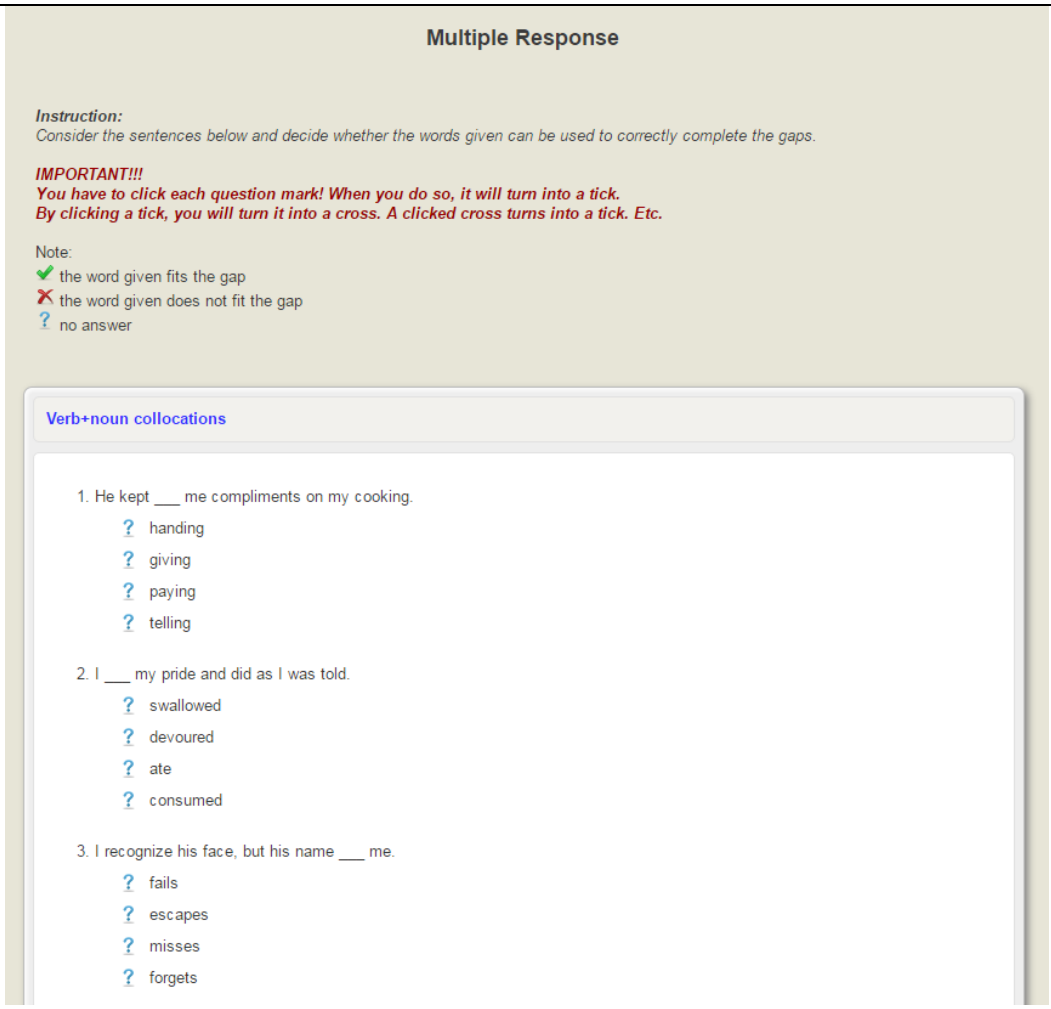
OR:

1. He kept [*handing] me compliments on my cooking.
2. [giving]
3. [paying]
4. [*telling]
5. I [swallowed] my pride and did as I was told.
6. [*devoured]
7. [*ate]
8. [*consumed]
9. I recognize his face, but his name [*fails] me.
10. [escapes]
11. [*misses]
12. [*forgets]

Source: Author

However, the way they are displayed for the test takers is quite different (Figure 6).

Figure 6: MR items (preview)



Source: Webclass (e) (n. d.)

Matching

A single MT item consists of a question (or premise) and a response (to be matched with the question). In the text converter, the questions are separated from the responses by means of an empty gap, indicated by square brackets (see Table 4).

Table 4: The formatting of MT items

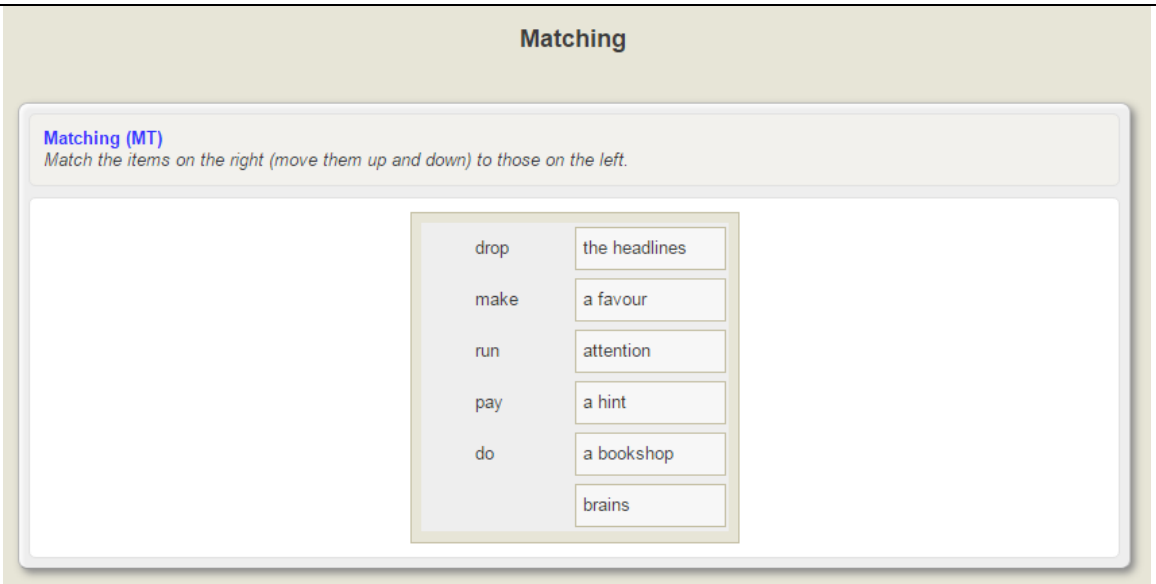
1. pay [] attention
2. run [] a bookshop
3. drop [] a hint
4. do [] a favour
5. make [] the headlines

Source: Author

The square brackets are used in this case for the sake of consistency; the parsing script can only proceed when a gap is found, and, as mentioned above, the item is created by saving the text preceding and following the gap.

In the main test editor, there are further options for MT items. For example, the test constructor may decide not to shuffle the questions (the responses are always shuffled). Moreover, to make the task more difficult for the test takers, one or more extra responses can be added (see Figure 7).

Figure 7: MT items (preview)



Source: Webclass (e) (n. d.)

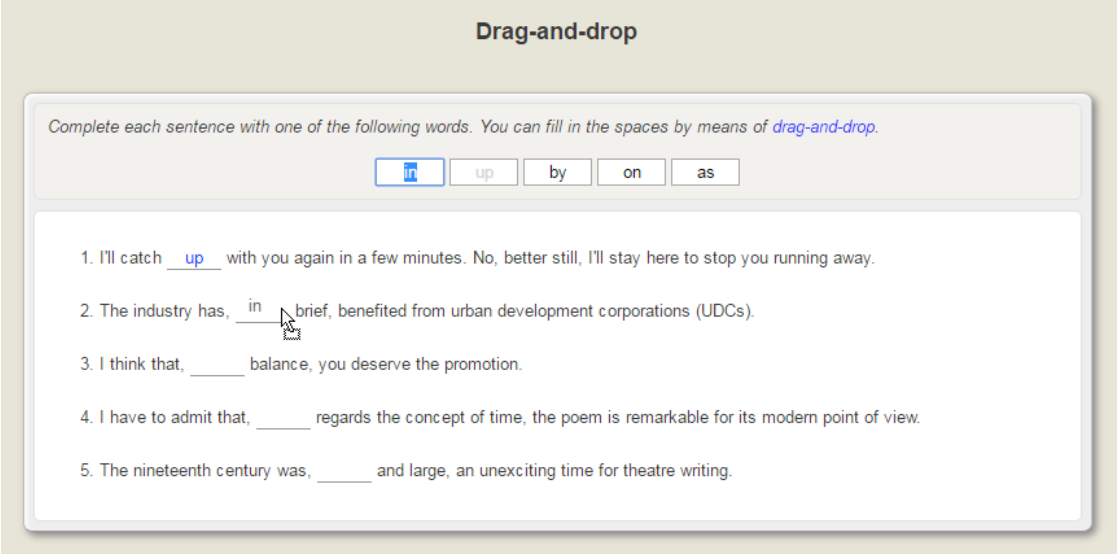
Other Selected-Response Formats

With standard gap-filling items (not discussed here), we expect test takers to produce their own answers. For example, the following sentences are to be completed with prepositions:

- They were sitting ____ the café.
- The cat is ____ the fence.

However, the words to be used in the gaps (responses) can be given in the rubric (instruction). This makes the task look more like matching than gap-filling, and definitely selected- rather than constructed-response. To make it even more "selected-like", test takers can be instructed to drag-and-drop the responses into the gaps, instead of typing them in. With the help of the text converter, the responses can be added to the rubric automatically simply by selecting the "Add answers for dragging" option. An example of such a task is given in Figure 8.

Figure 8: Drag-and-drop (preview)



Source: Webclass (e) (n. d.)

Conclusion

This article has taken an in-depth look at the principles and conventions of constructing selected-response test items with the aid of the text-to-items converter on WebClass. The converter works as an add-on to the main test editor, and its role is to accelerate the process of item writing. Using the text editor pane, an entire set of questions can be edited in the same way in which they are edited in a word processor. The parsing script then converts the text into test items and stores them in the database. In addition, the text converter can be used to automatically generate certain item types, for example, cloze passages. The converter also works in the opposite direction: it can retrieve existing items from the database and render them as formatted text displayed in the text editor pane.

References

Bachman, L. F., & Palmer, A. S. (1996). Language Testing in Practice: Designing and Developing Useful Language Tests. Oxford: Oxford University Press.

Bruno, J. E., & Dirkzwager, A. (1995). Determining the optimal number of alternatives to a multiple-choice test item: An information theoretic perspective. Educational and Psychological Measurement, 55, 959–966.

Douglas, D. (2010). Understanding Language Testing. London: Hodder Education.

Fulcher, G. (2010). Practical Language Testing. London: Hodder Education.

Gierl, M. J., & Haladyna, T. M. (Eds.). (2013). Automatic Item Generation: Theory and Practice. New York and London: Routledge.

Haladyna, T. M. (2013). Automatic item generation: A historical perspective. In M. J. Gierl & T. M. Haladyna (Eds.), Automatic Item Generation: Theory and Practice (pp. 13–25). New York and London: Routledge.

Haladyna, T. M., Downing, S. M., & Rodriguez, M. C. (2002). A review of multiple-choice item-writing guidelines for classroom assessment. Applied Measurement in Education, 15, 309–334.

Landrum, R. E., Cashin, J. R., & Theis, K. S. (1993). More evidence in favor of three-option multiple-choice tests. Educational and Psychological Measurement, 53, 771–778.

Malec, W. (2012). WebClass [learning management system]. Retrieved from http://webclass.co

Malec, W. (2015). Testing collocational knowledge: the question of item format. In A. Bloch-Rozmej & K. Drabikowska (Eds.), Within Language, Beyond Theories: Studies in Applied Linguistics (pp. 74–92). Newcastle upon Tyne, UK: Cambridge Scholars Publishing.

Parshall, C. G., Spray, J. A., Kalohn, J. C., & Davey, T. (2002). Practical Considerations in Computer-Based Testing. New York: Springer-Verlag.

Plannedparrothood (n. d.). Retreived from http://www.plannedparrothood.com/jokes.html

Roever, C. (2001). Web-based language testing. Language Learning & Technology, 5(2), 84–94.

Trevisan, M. S., Sax, G., & Michael, W. B. (1994). Estimating the optimum number of options per item using an incremental option paradigm. Educational and Psychological Measurement, 54, 86–91.

Webclass (a) (n. d.). Retrieved from http://webclass.co/tests_edit/import-from-text.php

Webclass (b) (n. d.). Retrieved from http://webclass.co/tests_edit/index.php

Webclass (c) (n. d.). Retrieved from http://webclass.co/tests/tests.preview.php

Webclass (d) (n. d.). Retrieved from http://webclass.co/tests_edit/import-from-text.php; the text in the editor area adapted from http://www.bbc.com/earth/story/20160412-what-really-happened-when-we-met-neanderthals

Webclass (e) (n. d.). Retrieved from http://webclass.co/tests/tests.preview.php