Behavior Research Methods
2007, 39 (4), 859-862

CheckVocal: A program to facilitate checking the accuracy and response time of vocal responses from DMDX

Athanassios Protopapas
Institute for Language & Speech Processing, Research Center "Athena," Maroussi, Greece

CheckVocal is a Windows application that facilitates checking the accuracy and response time of recorded vocal responses in naming and other experimental tasks using the DMDX display and response collection software. CheckVocal handles all keeping-track and presents each recorded response audiovisually (as waveform, spectrogram, and sound played out) along with the corresponding printed correct response and registered response time. The user simply decides whether the response was correct, wrong, or missing, with a single mouse click advancing to the next response. Response time correction can be done manually or automatically (retriggering by a power threshold). Data safety and integrity is ensured by cross-checking and status saving, so that interrupted sessions can be resumed later. CheckVocal is freely available to the DMDX community via a dedicated Web page.

Many psychological experiments require vocal responses to be recorded in response to some visually displayed or otherwise presented stimulus. "Naming" tasks, in particular, present a stimulus for the participant to pronounce its name, as for example in object naming, in which images of objects are displayed and the name of the object is the correct response. The Stroop (1935) task is perhaps one of the best-known color-naming tasks, in which color words are displayed with an incongruent font color—for example, the word "red" displayed with green letters, in which case the correct spoken response is "green," not "red."

DMDX[1] is a Win32 program that presents stimuli for psychological experiments and collects responses while measuring response time (RT). According to its creators, DMDX is "designed to precisely time the presentation of text, audio, graphical, and video material and to enable the measurement of reaction times to these displays with millisecond accuracy" (Forster & Forster, 2003, p. 116). DMDX is very simple to use and quite flexible in setting up and controlling many types of experiments, so there is a large user base for it and an active mailing list for help and support.

In addition to mouse clicks, keyboard presses, and external button-box responses, DMDX can collect and time vocal responses spoken into a microphone through the computer's audio card. Spoken responses are saved as individual audio files. Because it is so simple to set up a naming task with no special equipment other than a headset (or plain microphone) connected to the standard audio hardware, DMDX is a particularly good choice for researchers using such experimental tasks.

However, when working with naming tasks, there is an inevitable complication not present with other types of responses (e.g., using the mouse or a response button). The complication is that the experimenter must examine each and every response made by the experimental participants to check its accuracy. In the context of DMDX, this means that the experimenter has to listen to each individual audio file saved in the experimental run, and manually code whether the corresponding response is correct, incorrect, or missing. This is a very time-consuming process, and also an error-prone one, because the experimenter must take care to match the item numbers between the audio files and the response coding scheme.

In addition to determining response accuracy, which must necessarily be done manually, experimenters typically want to examine each recorded RT, to ensure that the voice-trigger mechanism has correctly registered it, because there are several sources of unreliability in automatic RT measurement. Too-high background noise, a transient event, nonspeech sounds made by the participant prior to the response (e.g., lip smacking, coughing, hesitation fillers, etc.), or late responses to the preceding items can cause RT triggering to register an incorrect measurement. Although it is possible to exclude some sources of timing errors by setting absolute thresholds (e.g., discarding response times below 100 msec or above a certain delay), it is not possible to ensure reliable response times entirely automatically.

CheckVocal is a Windows program written for the experimenter who uses naming tasks, aiming to facilitate the manual processing of spoken responses by automating all cross-checking and track-keeping, so that the experimenter will only have to check the accuracy of the response and its timing, using a convenient visual display. Thanks to multimodal presentation and optimized au-

A. Protopapas, protopap@ilsp.gr

tomatization, an experimenter can process more than a thousand trials per hour.

CheckVocal is written in Python, a powerful object-oriented scripting language,[2] with Tkinter, the standard Python interface to the Tk graphical user interface toolkit, and the Snack sound toolkit[3] for sound file handling, waveform and spectrogram display, and filtering, all of which are open-source free software.

How CheckVocal Is Used

DMDX reads information about the experiment from an RTF file ("item file"), in which each trial is described as an item with a numeric ID, along with timing, display, and other information controlling the experiment. DMDX saves its output data in a plain text file with .azk extension ("AZK file"), in which items (trials) are listed in the order presented, followed by the corresponding registered response time. These files are parsed by CheckVocal and the necessary information is extracted. In addition, the experimenter must also provide the correct responses in a separate text file, with a filename starting like the item but ending with "-ans.txt". For example, for experiment Exper1.rtf, the answers should be in Exper1-ans.txt. Let us assume that Exper1.rtf is a naming experiment in which the words "cat" and "dog" appear on the screen in two separate trials, and that the correct responses to these strings are "yes" and "no," respectively. Then Exper1.rtf will contain the following item specification,

```
+1 * "cat" / ;
+2 * "dog" / ;
```

and Exper1-ans.txt must contain these lines to specify the corresponding correct responses:

```
1 yes
2 no
```

Figure 1 shows a screen shot from the main CheckVocal window, which displays the correct response ("pint") on the top row and the waveform and spectrogram of the recorded response below it. CheckVocal displays each waveform and spectrogram along with the corresponding timing mark that was registered by DMDX in the course of running the experiment. At the same time, CheckVocal plays out the recorded response; not from the beginning of the file but from the RT mark onward, so that the experimenter can verify that the beginning of the spoken response is not lost.

If the experimenter finds that the RT mark is inappropriately placed, he or she may click on the waveform or spectrogram to manually place the timing mark, or request that CheckVocal automatically retrigger to calculate the RT on the basis of an adjustable threshold. For cases in which an intruding event (lip opening, filler, etc.) precedes the actual response, CheckVocal can automatically detect the next onset following a silent interval after the current RT mark. Every time a response waveform is displayed or a timing change is made, the sound is played out automatically from the timing mark on. To further verify timing, the experimenter may, if desired, click to hear the sound up to the timing mark (play left), or to zoom into the waveform in higher temporal resolution.

For each displayed waveform, the experimenter must indicate, with a single mouse click, whether the spoken response is correct or wrong, or whether there is no response. This is done rapidly on the basis of comparing the printed correct response with the auditorily presented recorded response. The response to the next item (trial) is then automatically loaded and immediately displayed and played out. Thus, if the voice-trigger threshold (either from DMDX or in CheckVocal) is properly adjusted (or if the experimenter is not interested in the response times), going through the responses can be extremely fast, with a single click per trial.

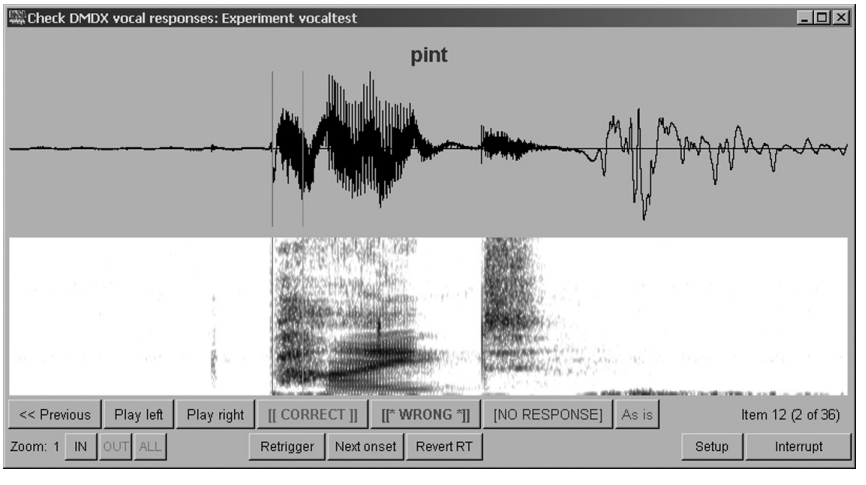

Figure 1. The main response viewing window of CheckVocal, with the correct response displayed in the top row, the recorded spoken response displayed as a waveform and the corresponding spectrogram, the rating and listening buttons in the row under the spectrogram, and the auxiliary function buttons (zoom, trigger) in the bottom row. The current timing mark appears as a bright red line over the waveform and spectrogram, and a faint gray line marks the original response time registered by DMDX.

CheckVocal shows a progress indicator and the total number of responses to be examined. Backward and forward navigation through the responses allows checking and correction of mistakes, if necessary. The process can be interrupted at any time, and then optionally resumed later simply by rerunning CheckVocal and selecting the same AZK file for processing.

The output of CheckVocal is saved in a plain text file with a filename starting like the experiment (item) file, but ending with "-datalist.txt". By default this is a tab-separated file of RTs, one line per subject, including a first row of column headings (with the item numbers). Following DMDX convention, incorrect responses are indicated with negative RT values, and no-response trials are indicated with negative values equal to the timeout parameter of the experiment. This output file from CheckVocal is trivial to load directly into a spreadsheet or statistical program for further processing.

How CheckVocal Operates

CheckVocal is composed of a few main objects and several secondary objects and functions. One object holds parameters that need to be globally accessible. Another implements the graphical interface shown in Figure 1, using Snack sound objects and canvas items, and responds to the user's actions. A third main object handles the linear procedures of data parsing, status updates, and output, launching interactive panels when needed.

CheckVocal makes heavy use of sequence objects offered in the Python language, in particular dictionary and list objects. Lists are used to hold participants, trials, files, and correct responses; dictionaries hold indices of participant IDs and trials, as well as dates and numbers. The flexibility and convenience offered by these constructions offered by Python cannot be overemphasized.

In brief, CheckVocal operates in the following sequence: As soon as a root window is initialized and withdrawn, SetupWindow collects the user's selections, notably the AZK file to process. A CheckVocal object then parses the AZK file and sets up lists of participants and trials, validating each component along the way. The RTF file is parsed to determine the time-out value, and the correct responses are read from the ANS file and verified against the available item list from the participant data. Then the existence of all needed response audio files is verified.

If a previous unfinished session is detected, the user is given the option to continue or start over. The list of responses to check is passed on to CheckWaves, which presents the graphical interface with the waveform, spectrogram, and action buttons, and modifies RT values accordingly. When CheckWaves returns, the original order of participants in the experiment is reconstituted, as determined from the AZK file, and the output is saved along with the log file.

Parameters and Adjustments

When CheckVocal is started, a setup panel is displayed (see Figure 2), on which the experimenter may set various options. Selection of an AZK file to process responses from is mandatory. The folder in which the selected AZK file was loaded from is saved in the system registry so that it can serve as default location for the next run of the program. All that is necessary to run CheckVocal is to select an AZK file and to click on "Proceed." However, a number of options are provided for additional flexibility.

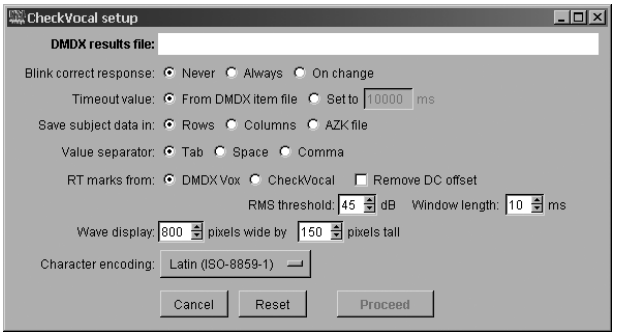

Figure 2. The initial setup screen of CheckVocal, with all the adjustable parameters at their default settings.

When each item in an experiment has a different correct response, the displayed response string changes with every audio response, and it is not difficult to check immediately for accuracy. If, however, most responses are of one of a few types (e.g., three or four color names in a Stroop task), most audio responses will be the same, as will most correct response strings. In this situation, it is easy to stop paying attention to the fixed response string displayed on the screen and thus to possibly miss an incorrect response. To help avoid this situation, the option is given to blink the displayed response word so that it attracts the experimenter's visual attention. Perhaps the most useful option is to blink "on change"—that is, whenever there is a change of correct response, which is when attention is needed. The option to blink "always" is also available.

To signify a missing response, CheckVocal uses the time-out value determined from the item file, in accordance with DMDX convention. However, should the experimenter prefer a different value to serve as "missing" flag for the subsequent statistical processing, this can be set as manually specified "time out."

The default output option of one row per subject, with tab-separated items, is reasonably convenient for most statistical processing programs. However, alternative output formatting is available—for example, space-separated or comma-separated instead of tab-separated items, and subject columns instead of subject rows. In addition, the output can be saved in AZK format, if further processing with other AZK-reading software is required.

Vocal response timing typically relies on the DMDX Vox (voice trigger). If the Vox was not properly adjusted when the experiment was run, response times will be inaccurate, and most likely unusable. DMDX provides an offline voice trigger option, to automatically recalculate RTs given a power RMS threshold, which is adjustable from this panel. The duration (window length) over which power is calculated is also adjustable. Power calculation depends on a proper zero level, but cheap microphones and poor audio cards providing no biasing current in most

common equipment result in a fluctuating or offset zero level in the recordings. The option to remove DC offset corrects this problem, so that voice triggering works properly.

The size of the display window can also be adjusted to suit recorded response duration, personal preferences, and individual circumstances. Finally, an encoding option is offered, currently supporting Greek response strings in addition to Western (Latin) ones. More encoding options can be added for users requiring different response encoding who are willing to provide test materials and help with testing.

Data Safety

One very important feature of CheckVocal concerns the safety and integrity of the experimental data. Because the main goal of the program is to facilitate a common procedure and speed up the process, any lack of reliability or sensitivity to failures (of power, hardware, or operating system) would defeat its purpose. Taking into account the total number of responses for a naming experiment with a few tens of participants and a few hundred responses from each of them, it is of crucial importance to ensure that any interruption of the response-checking procedure does not invalidate work already done. For this reason, CheckVocal saves its status completely after every single response, flushing all of its file buffers, so that data loss may be avoided even in the case of a catastrophic failure.

During operation, a "console" window is present on the screen, on which verbose status messages are reported, to facilitate corrections in case something goes wrong. A log file is also created. In addition to program status messages, all modifications made to response times are listed both on the console window and in the log file. Of course, no alteration is made to the original AZK file produced by DMDX in the course of running the experiment. Thus, the experimenter has access to all stages of data processing and can compare the reliability of the procedures or of voice triggering in different conditions and situations.

Data integrity is enforced at all stages of processing. When item files are parsed, every participant's item IDs are checked against a reference item list and against the list of correct responses. Any discrepancies in labels, missing or incomplete data lines, or missing audio files result in deactivation of the corresponding participant (with a message to the user). When reading in previously saved status files in order to resume processing, everything is checked once more, including the existence of audio files and the matching between status components, even among those saved by CheckVocal itself, which are not normally expected to have been altered. In this way, the experimenter can be reasonably confident in the reliability of the data in the resulting files.

The output of CheckVocal (including data list, change log, and subject selection) is always saved in the same folder as the working data file (and the accompanying item and answer files as well). This way, all results and processing associated with the data will be in the same place and will run no risk of being overwritten (e.g., by the same experiment run later with a different group of participants). For data safety, the output will not overwrite an existing file unless confirmed by the user. So, if a -datalist.txt file for the same experiment already exists in the current working folder, the experimenter will have the option to specify a different output file name (and/or folder).

Obtaining and Installing CheckVocal

CheckVocal is free, open-source, public-domain software, available from the author's Web site at www.ilsp.gr/homepages/protopapas/checkvocal.html. A self-extracting executable can be downloaded that contains all necessary files and libraries. Installation amounts simply to extracting the contents of the downloaded archive to a convenient folder. Experimenters who plan to use CheckVocal a lot may wish to create a shortcut to CheckVocal.exe on their desktop or quick-launch toolbar. The source code (Python script) is also available from the same Web page, for experimenters with Python and Snack already installed on their computers.

AUTHOR NOTE

The author thanks Jonathan Forster, for permission to use and modify the DMDX icon; Kåre Sjölander, for providing the Snack DC filtering solution; Fredrik Lundh, for "An Introduction to Tkinter," freely available on the Web at effbot.org/tkinterbook/; Markus Damian, Gareth Gaskell, Kathy Rastle, and Michael Reynolds, for providing valuable feedback on an early version; and Eleni Vlahou and Persefoni Bali, for continuous tireless testing. Correspondence regarding this article may be sent to A. Protopapas, ILSP, Artemidos 6 & Epidavrou, GR-15125 Maroussi, Greece (e-mail: protopap@ilsp.gr).

REFERENCES

FORSTER, K. I., & FORSTER, J. C. (2003). DMDX: A Windows display program with millisecond accuracy. *Behavior Research Methods, Instruments, & Computers*, **35**, 116-124.

STROOP, J. R. (1935). Studies of interference in serial verbal reactions. *Journal of Experimental Psychology*, **18**, 643-662.

NOTES

1. www.u.arizona.edu/~kforster/dmdx/dmdx.htm.
2. www.python.org.
3. www.speech.kth.se/snack/.

(Manuscript received July 24, 2006;
revision accepted for publication December 27, 2006.)