LMEDS 2.0 Instruction Manual

Tim Mahrt

August 5, 2015

It is my hope that you will find this document useful in working with LMEDS. This document contains information for all aspects of LMEDS, from installation and setup, to creating experiments, running experiments, and obtaining and analyzing test results. If you have any questions, comments, or other feedback, please contact me at timmahrt@gmail.com

Contents

1 User-Defined Test Components

An LMEDS test has the following components

1.1 Audio files

Audio file can be any common type (.mp3, .wav, etc.). Some web browsers are unable to play audio in certain file types. Personally, use both .mp3 and .ogg

See this page for more information on browser compatibility: http://www.w3schools.com/tags/tag_au

Audacity makes it very easy to convert entire directories of audio files into multiple audio formats.

1.2 Transcription files

The transcript for an audio file needs to have the same name as the audio file, except for the extension (e.g. the transcript for "water.wav" should be "water.txt"). Only raw txt files with the extension .txt are accepted.

For long excerpts, you should specify the line breaks explicitly. For a long excerpt, if all of the text lies on one line, the text will run off the page in LMEDS.

1.3 Sequence file (see section 4)

The file that specifies the control flow of a test. It specifies what instructions and stimuli users will be presented with in a test and in what order.

Multiple sequence files can exist for a set of audio and transcription files (e.g. one could have a sequence with the stimuli in a certain order and another sequence with the stimuli in a different order or with different instructions, etc.).

1.4 Dictionary file(s) (see section 2)

The file that contains all text that will be seen by users in your experiment. This file is independent of a sequence file"so one could write a sequence and present it in both English and Russian.

1.5 Survey file(s) (see section 3)

These are optional file(s). A survey file can be used to get various kinds of feedback, such as might be needed for a longform survey or a short-answer question. Supports text boxes, checkboxes, and radio buttons.

Each "survey" requires its own survey file (e.g. presurvey.txt, postsurvey.txt).

The difference between a survey and a regular page in LMEDS (which was defined in python, such as a boundary annotation page) is that a survey page is fairly constrained in how the items are rendered while a non-survey page is very flexible but its creation is significantly more involved.

1.6 .CGI file

This is a file that currently the LMEDS administrator maintains and creates. It is used to "glue" together all of the components that make up a test (sequence file and language file). It is the only part of a test that is directly accessible to users.

This is not a file that you normally need to create, unless you are the LMEDS administrator. A template of a typical .cgi file can be found in the section 7.

2 Dictionary File Specification

LMEDS uses "dictionaries" for multi-lingual support. A set of descriptive unique keys are created and used (unchanged) across each language. When that key is used in a page, LMEDS looks for the corresponding text for that key in the target language. E.g. the key "dictionary" will fetch the text "Dictionary" for English and "Dictionnaire" for French. Each dictionary file is created by the user.

Some keys are defined within the python code. Other keys are user-defined. Eventually a tool will exist that will generate a template dictionary file for a sequence. Right now, the best way to create a custom dictionary is to start with a dictionary file used in a prior test.

A dictionary files has the following components (a complete example follows these):

2.1 Sections

A section is denoted with "-" characters on the line above and below the section name.

Sections are actually ignored in the code. Their only purpose is to help structure dictionaries"perhaps most naturally by page.

2.2 Keys

A key is denoted in the same with "=" characters. Keys must be the same, regardless of the language.

2.3 Texts

A "text" appears after a key and before another key or section. Unlike sections and keys, text is undecorated text.

Text is rendered in HTML. This means HTML markup is allowed. It also means that whitespace other than a single space is ignored. If you want a line break, for example, you will need to insert one used ¡br /¿

Here is a short, concrete example

———— login page ————

========== progress ==========

Progress

========== title ==========

Perception of Spoken Discourse

The section denotes that these keys are used in the login page. Whenever LMEDS needs to display the title, it will look at the text for the key "title""in this case "Perception of Spoken Discourse".

3 Survey file specification

The survey format allows for the simple creation of short or long questionnaires. The format is simple-the first line in a survey question contains the text prompt, subsequent lines specify one or more data entry fields that users can use to answer the question, and, finally, a blank line signals that the current item is complete (an example is at the bottom of this section).

A note on arguments: For data entry fields that have arguments, arguments should be separated from the data entry field name by a space and from each other by a comma e.g.

Choice English, Arabic, Other

Here is the list of data entry fields available:

3.1 Choice

Users can select exactly one item out of many.

e.g. Sex: Choice Male, Female

3.2 Item_List

Users can select as many items as they want out of many.

e.g. Indicate the language(s) that you are familiar with. Item_List English, French, Spanish, Italian, German

3.3 Choicebox

A dropdown box. Users can select one item from many. Similar to a Choice but is more efficient in space for large lists of items.

e.g. Level of education completed: Choicebox High School, Some College, Bachelor's Degree, Master's Degree, PhD

3.4 Textbox

A single line for users to enter a small amount of information. A textbox takes no arguments.

e.g. Occupation Textbox

3.5 Multiline_Textbox

A longer-form textbox that can span several lines. It takes exactly two arguments: the number of characters across and the number of lines.

e.g. What did you think of this test? Please provide any feedback. Multiline_Textbox 50, 7

There is one more feature for creating surveys. It is possible to designate a group of items as subquestions of the previous question. These items will be displayed tabbed and have a new numbering scheme. To do this, encapsulate the relevant items in the following tag: ¡sublist¿ ¡/sublist¿

Here is a short concrete example demonstrating the key features discussed above

Sex: Choice Male, Female

Age: Textbox

Country of birth: Choice United States, Other Textbox

¡sublist¿ If United States, list city/state: Textbox

If other, how old were you when you moved to the USA? Textbox ¡/sublist¿

Thus we have a survey with 5 items in total (2 being subitems to item 3). Item 3 shows how multiple data entry fields can be placed on a single question (if the user selects "other" for the first question they are expected to specify what they meant in the Textbox).

4 Sequence file specification

The sequence file contains the items that will be presented in a test. The first line in a test is the test name which should be prefixed with a "*" (e.g. *My_Test_Sequence). The second item must be "login" where users create a name to associate their data with. If login is not the item on the second line, unexpected behavior can result. Subsequent items are presented in a linear fashion as users progress through the sequence. The last item in the sequence must be "end".

Most pages take arguments. Arguments are separated from one another and from the page type by a space. Audio files and textfiles included in an argument should not include their file extension (.txt, .wav, etc.)"LMEDS will determine the appropriate extension to be used. An optional feature that can be used with arguments is found at the bottom of this section.

Here is a list of page types currently supported, broken into two sets: administration and stimuli presentation. See the bottom of this section for a complete example of a sequence file.

4.1 login

This is the first page that users see. They enter their name here. Names must be unique. If someone has already attempted to start a test under that user name the user will not be able to proceed from the login page.

"login" takes no arguments.

4.2 audio_test

Some users have reported an inability to hear audio in LMEDS"this is usually related to a particular browser on a particular computer. For this reason there is the audio_test page. This page can be presented right after a user logs in to save them time in the event that their browser is not correctly playing audio.

"audio_test" takes a single audio file as an argument. I recommend using a short pure tone.

e.g. audio_test test_tone

4.3 consent

Presents the consent form. If users opt not to consent, the test ends immediately. If they consent, they proceed to the next page.

"consent" takes one argument which specifies the consent form to display to users (the consent form is contained within the language dictionary)

e.g. consent main_consent

4.4 text_page

A page that displays nothing but the text from a single text key. This page could be used to provide task instructions to users, indicate that they should take a break, etc.

Unlike other pages, text_pages are interpreted as only having a single argument, even if the key contains spaces.

e.g. text_page first block instructions

4.5 survey

Specifies a survey page (see Section 3 for info on surveys).

"survey" takes a single argument, the file name of the survey that it should load

e.g. survey presurvey

This would load the survey called "presurvey.txt"

4.6 end

The final page of the test.

"end" takes no arguments.

Here is a list of page types involving stimuli presentation

4.7 prominence

Users are presented with an audio file and the associated transcript. They can click on a word to indicate that it is prominent. This changes the selected word to "red".

"prominence" takes the following arguments: - audioFileName " the name of the audio file - minPlays " the minimum number of times the audio file has to be played before the user can continue - maxPlays " the maximum number of times the audio file can be played before the audio button is disabled - instructions - the type of instructions to show (e.g. meaning vs acoustics) (default None " no special instructions) - presentAudio " either "true" or "false", specifies whether the audio is hidden or presented (default True). If "false" the values for "minPlays" and "maxPlays" are ignored.

e.g. prominence water 1 3 acoustics true

4.8 boundary

Users are present with an audio file and the associated transcript. They can click on a word to mark the presence of a boundary after it. This places a solid line after the word.

"boundary" takes the following arguments: - audioFileName " the name of the audio file - minPlays " the minimum number of times the audio file has to be played before the user can continue - maxPlays " the maximum number of times the audio file can be played before the audio button is disabled - instructions - the type of instructions to show (e.g. meaning vs acoustics) (default None " no special instructions) - presentAudio " either "true" or "false", specifies whether the audio is hidden or presented (default True). If "false" the values for "minPlays" and "maxPlays" are ignored. - boundaryToken " can use a token other than the default. (default None " the default value is not a font character, but a border between words).

e.g. boundary water 1 3 acoustics true &

4.9 boundary_and_prominence

A combination of A) and B). Users first mark boundaries. They then can mark prominences. While marking prominences they can see but not change the boundaries that they marked"this is the only difference between this page and splitting the prominence and boundary task across two pages.

The arguments for boundary_and_prominence are the same as for boundary pages.

e.g. boundary_and_prominence water 0 3 acoustics true

4.10 same_different

The user is presented with two audio files and has to decide whether they are the same or different.

"same_different" takes the following arguments - audioName1 - audioName2 - minPlays - maxPlays

e.g. same_different water1 water2 0 3

4.11 axb

The user is presented with three audio files: A, X, and B. They have to decide if X is more like A or more like B.

"axb" takes the following arguments - sourceNameX " the name of the audio file for X - sourceNameA " " for A - sourceNameB " " for B - minPlays -maxPlays

e.g. axb water1 water2 water3 1 2

4.12 abn

The user hears one audio file and decides if it is an example of text prompt A, B, or neither A or B.

"abn" takes the following arguments - audioName - minPlays -maxPlays

e.g. abn water 0 2

A Note on Default Values

This is a completely optional feature. If you find it confusing you can skip it.

Some page types, such as "prominence" have default values for some arguments. An argument might have a default value if alternative values are uncommon or to support new functionality without requiring changes to pre-existing code.

Arguments that have default values do not have to be specified in a sequence file if you are ok with the defaults"this can make your sequence file cleaner and easier to read and maintain.

There are two ways that you can specify an optional value. One is to list it as normal e.g. prominence water 1 3 acoustics true

A second way is to refer to it explicitly by its name as described above. e.g.

prominence water 1 3 instructions=acoustics presentAudio=true

This second way is useful, for example, if you wanted to specify the value for a later variable such as "presentAudio" but not "instructions". The only way to do this is by naming the variable

Prominence water 1 3 presentAudio=true

If one attempted

Prominence water 1 3 true

This would set "instructions=true" which would cause an error unless instructions named "true" existed in the dictionary file.

5 Parsing the LMEDS output

The output to LMEDS is quite simple. Each line has 4 components:

- the page type
- the argument list
- peripheral information collected on the page (number of times users listened to audio files, time spent on the page, etc.)

- the users input in data-entry fields presented on the page

For most needs, A and D contain the most important bits of information. A is separated from the rest of the row by the occurrence of the first comma. D is separated from the rest of the row by the sequence ";,". If you need something from B or C, you will need to parse it"B and C both vary depending on A.

D is a string of comma-separated values. Every individual item in a data-entry field, except for textboxes, is represented by a 0 or 1, where 0 indicates the user did not choose that item and 1 indicates the user did choose that item. So if the user has a choice between A or B and selects A, their output will look like 1, 0. And if they select B: 0, 1. For a boundary_and_prominence page, each word gets two digits in the output (one for boundary and one for prominence) so the output string can easily contain over 100 values.

Let"s look at a real example. Here we have the sequence:

same_different water water 1 -1 axb water water water 0 2 abn water 0 2

And here is one possible output for it

same_different,water;water;1;-1,0,0,0:4.8;,1,0 axb,water;water;water;0;2,0,0,0:6.7;,0,1 abn,water;0;2,0,0,0:3.3;,1,0,0

If we separate out just A and D we can easily see the data that we want to analyze:

same_different;,1,0 axb;,0,1 abn;,1,0,0

LMEDS offers some useful post-processing that can help prep data for analysis. Otherwise, with the above information, you should have everything you need to do your own analysis.

6 Data post-processing

Aggregation and analysis is an ongoing development. The data generated by LMEDS is user-centric"each user has one file with all of their data in that file. However, researchers are often not interested in this user-centric view but a stimuli-centric view"how does a user"s data compare to other users for a given stimuli item. LMEDS can automatically aggregate and transform survey data and prominence and boundary data to have this stimuli-centric view"where all data for a given stimuli is stored within one file.

LMEDS also can automatically calculate various statistical measures of the data. However, this is currently one of the least developed aspects of LMEDS.

7 Creating your own LMEDS installation

LMEDS is very quite easy to install for someone working in an IT department but will be quite difficult for someone not familiar with setting up and maintaining a server. That said, plans are underway to further automate the process and to simplify the installation of new tests.

Setting Up the Server

If you are installing a server from scratch, you"ll need to make sure it comes with hooks to python"which is fortunately fairly typical these days. In your httpd.conf file you"ll want to enable .cgi files. That is all! LMEDS only uses standard python libraries and is based on python 2.7 (although I don"t believe any 2.7-specific features are being used).

Setting Up the Directory Structure

Under the root page of LMEDS are the following folders:

7.1 cgi-bin

In the source code is a subfolder called /lmeds/"dump the contents of that folder into cgi-bin

Create a .cgi file for all tests that you want to run (see below for a template .cgi file). Make sure the tests are executable.

7.2 html

Contains a few shared html templates and snippets. Copy these over from the source code folder /resources/html/

7.3 imgs

Contains a few shared images. Copy these over from the source code folder /resources/imgs/.

7.4 tests

Under /tests/ you will create a subfolder for each dataset that you are using. Experiments being run with over the same data can be run from the same subfolder but with different sequence files (to provide different instructions or different order to stimuli items, etc.).

The first line of every sequence file contains a name prefixed by a * character. Create a subfolder under /tests/¡¡Test name¿¿/output/ with the name given in that first line. Change the file permissions to read and write (but not execute!) for everyone.

Congratulations! You are ready to begin running subjects!

8 CGI Template

""" #!/usr/bin/env python # -*- coding: utf-8 -*-
import cgi import cgitb cgitb.enable()
import rptMain
if __name__ == "__main__": # disableRefreshFlag=False is useful when you want to debug a problem survey = rptMain.WebSurvey("name_of_folder_inside_subfolder_¡tests¿", "sequence_name.txt", "language_dictionary_name.txt", disableRefreshFlag=True) survey.run() """

9 LMEDS Change History

August 2015 - LMEDS 2.0

- First public release.
- Inclusion of user script utilities.
- Numerous bugfixes and stability improvements (audio is significantly less error prone).
- Offline server for running experiments locally.
- Companion website.

May 2014 - LMEDS 1.5

- Object oriented refactor.
- Numerous bugfixes and stability improvements.

November 2013 " LMEDS 1.0

- First official release.

- First official experiments run using LMEDS.

May 2013 ” LMEDS 0.0

- Work begins on LMEDS

10 LMEDS Manual Change History

August 5, 2015
Second release. Move from Word to LaTeX for easier tracking of changes.
December 17, 2013
First release