Turkshop: technical parts

mitcho + Hadas

McGill Graduate Seminar
February 2015

In these slides

- The relationship between templates and Turk item files
- Examples of different kinds of experiments
- Preparing for Turk (technical workflow)

Some terminology

- An **item set** is a set of sentences/stimuli that vary the factors of interest in a systematic way, and hold constant everything else.
- A **condition** is a label for a particular setting of all of the factors of interest.
- Individual stimuli within an item set are called **items**.
- Items are grouped into **sections**. Normally, “target” and “filler.”

```
# blocking 1 inanimate-make-v
That’s the ball that the coach bounced on the floor.

# blocking 1 inanimate-v
That’s the gymnast that the coach bounced on the floor.

# blocking 1 animate-make-v
That’s the ball that the coach made bounce on the floor.

# blocking 1 animate-v
That’s the gymnast that the coach made bounce on the
floor.
```

Getting set up

Instructions (for later):

- Install Python 2.7.3: `http://www.python.org/getit/` (choose 2.7.3, not 3.x)
- Download turktools: `http://turktools.net`

For today:

Download files from: `http://people.linguistics.mcgill.ca/~michael.erlewine/turk/`

Your template and items

What people see:

In order to get paid, please make sure that you answer all 14 items.

Consent Statement: By answering the following questions, you are participating in a study being performed by linguists in the Department of Linguistics and Philosophy at Massachusetts Institute of Technology. If you have questions about this research, please contact EMAIL. Your participation in this research is voluntary. You may decline to answer any or all of the following questions. You may decline further participation at any time without adverse consequences. Your anonymity is assured; the researchers who have requested your participation will not receive any personal information about you.

---

1. That's the sentence that the linguist added without thinking.

NATURAL ○ UNNATURAL ○

---

2. That's the idea that the team floated at the meeting.

NATURAL ○ UNNATURAL ○

---

3. That's the rabbit that the magician made vanish into thin air.

NATURAL ○ UNNATURAL ○

---

4. That's the intern that the supervisor returned for another shift.

NATURAL ○ UNNATURAL ○

Your template and items

What you give Turk:

Template file
`.html`

Turk items file
`.turk.csv`

☞ Open up `binary-mcgill-TK1-10.html` in your browser. Probably double-clicking on it will work.

- What kind of experimental paradigm is this template for?
- How many items is this template file expecting?
- How is it different than what the subject sees?

Your template and items

What you give Turk:

Template file
`.html`

Turk items file
`.turk.csv`

☞ Open up `binary-mcgill-TK1-10.html` in your browser. Probably double-clicking on it will work.

- What kind of experimental paradigm is this template for?
- How many items is this template file expecting?
- How is it different than what the subject sees?

Your template and items

What you give Turk:

Template file
`.html`

Turk items file
`.turk.csv`

☞ Open up `binary-sample-items.turk.csv` in Excel. (CSV files are a kind of plain-text spreadsheet.)

- How many separate lists does this include?
- How many items do these lists have?
- What's the relationship between list 0 and 1? 2 and 3?

Your template and items

When you upload a template and item file to Turk, each one of these lists will be turned into a HIT (“Human Intelligence Task”).

Each string in that row in the Turk items file will be plugged into “fields” in the template.

list	trial_1_1	trial_2_1	⋯
0	That’s the rabbit that the magician made vanish into thin air.	That’s the boy that the instruc-tor made float in the pool.	⋯
⋮	⋮	⋮	⋱

In list #0, the text `${trial_2_1}` in the template file will be replaced with “That’s the boy that the instructor made float in the pool.”

Your template and items

When you upload a template and item file to Turk, each one of these lists will be turned into a HIT (“Human Intelligence Task”).

Each string in that row in the Turk items file will be plugged into “fields” in the template.

list	trial_1_1	trial_2_1	⋯
0	That’s the rabbit that the magician made vanish into thin air.	That’s the boy that the instructor made float in the pool.	⋯
⋮	⋮	⋮	⋱

In list #0, the text `${trial_2_1}` in the template file will be replaced with “That’s the boy that the instructor made float in the pool.”

Experiment types and skeletons

We create templates from “skeleton” files. Think “template recipes.”

Each skeleton corresponds to a different kind of experiment. Let’s open some skeletons in our browser.

- `binary.skeleton.html`
- `binary-image.skeleton.html`
- `completion.skeleton.html`
- `image-choice.skeleton.html`

More described in the paper.

See also: audio playback example.

Experiment types and skeletons

We create templates from “skeleton” files. Think “template recipes.”

Each skeleton corresponds to a different kind of experiment. Let’s open some skeletons in our browser.

- `binary.skeleton.html`
- `binary-image.skeleton.html`
- `completion.skeleton.html`
- `image-choice.skeleton.html`

More described in the paper.

See also: audio playback example.

The Lister

The Lister does the following:

- Create Latin Square lists from the items, so that each participant sees only one condition per item.
- Randomize the lists and mix the targets and fillers together, following certain constraints.
- Print the resulting lists into a Turk items file (`.turk.csv`).

The Lister input file

☞ Open `binary-image-sample-items.txt`, which is a Lister input file.

Each *trial* has a *header* beginning with #. The header includes a section name, item number, and condition name.

```
# target 1 most
Most of the dots are blue.
http://web.mit.edu/hkotek/www/experimentpictures/
gradientYN/2C-1.png

                                    ⋮
```

Each line of the trial corresponds to a different *field* in the template. For this experiment, each trial has a sentence and a URL of an image. At least one blank line is required between trials.

☞ How many sections are there? How many items do they have? How many different conditions are there?

☞ How many items will an individual participant be asked to answer?

The Lister input file

☞ Open `binary-image-sample-items.txt`, which is a Lister input file.

Each *trial* has a *header* beginning with #. The header includes a section name, item number, and condition name.

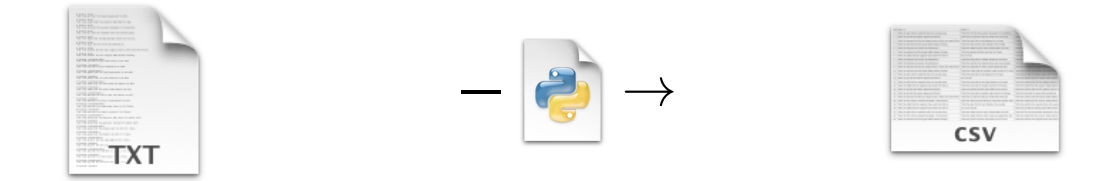
```
# target 1 most
Most of the dots are blue.
http://web.mit.edu/hkotek/www/experimentpictures/
gradientYN/2C-1.png

                                    ⋮
```

Each line of the trial corresponds to a different *field* in the template. For this experiment, each trial has a sentence and a URL of an image. At least one blank line is required between trials.

☞ How many sections are there? How many items do they have? How many different conditions are there?

☞ How many items will an individual participant be asked to answer?

Creating lists for Turk



Lister input `.txt` | `lister.py` | Turk items `.turk.csv`

☞ Run `lister.py`. Enter `binary-sample-items.txt` and request a multiple of 2 lists. It will ask you about filler placement constraints.

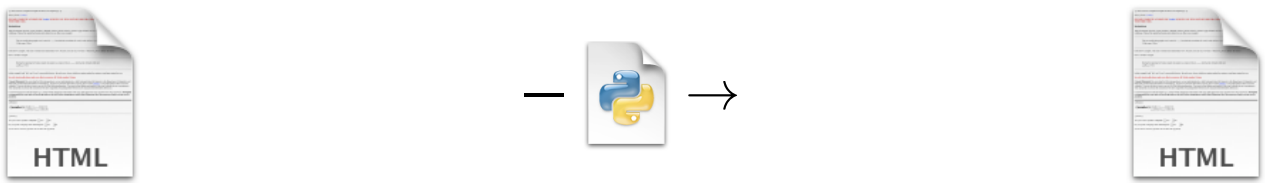
It will then print a nice summary of how many sections, items, conditions, etc. were found and (hopefully) tell you that the CSV files were successfully created.

NB: Double the number of lists you request will be created. That's because the reverse order of each list is also always included, to attempt to counter ordering effects.

☞ How many items does it say are in each list?

`templater.py`

We need a template of the right length. We have a tool to create templates of arbitrary length from skeletons:



template skeleton — `templater.py` → template

☞ Run `templater.py`. Create a template based on `binary-image.skeleton.html` for 108 trials. Pick a code, any code.

Open the resulting template file and verify that it's built for 108 trials.

Putting it together

☞ Use `simulator.py` to put your new template and our new Turk items file together. Make sure the resulting simulation looks good.

template Turk items `simulator.py` simulation

Now you have a `binary-mcgill-TK1-10.simulation.html` file in your folder. Look at it in your browser. Note that Turk will add a submit button, so our template doesn't need to include that.

At this point you would be ready to upload your template and Turk items (`.turk.csv`) file to Turk!

Turk technical workflow overview

aka "Turkflow"