Human Evaluation of NLP System Quality

INLG Tutorial, 24th September 2024

Unit 6: Experiment Implementation

Link to Unit 6 Resources

Overview

Unit 6: Experiment Implementation

Overview

Unit 6: Experiment Implementation

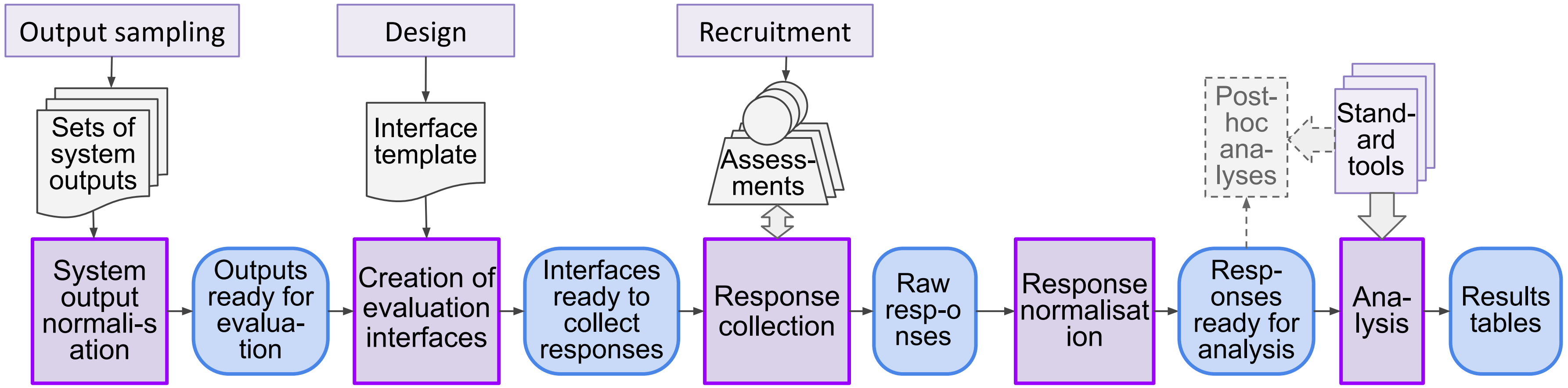
Unit aims and learning outcomes

- The aims of Unit 6 are:
  - To look at the specification for each component process from the experiment design (Phase II, Unit 4) and implement it as code where possible.
  - To cover the pipelining of component processes and documentation of the implementation.
  - To discuss good gooding practices.
- After completion of the unit, participants will be able to:
  - Implement a given design for a human evaluation experiment such that it is automated and repeatable.
- Prerequisites: Units 2–5.

Overview

Unit 6: Experiment Implementation

Component process diagram for reference



The 11 design steps and outputs from them

Step	Description	Outputs from Phase I (Design) that feed into implementation	Process as part of which implemented
1	Systems; Quality criteria & evaluation modes	N/A	N/A (see Design)
2	Number of outputs & evaluators	a. Number of outputs	Output sampling
		b. Number of evaluators	Recruitment
		c. Assignment of output to evaluators	Creation of interfaces
3	Output sampling & normalisation	a. Specifications for output sampling	Output sampling
		b. Specifications for output normalisation	System output normalisation
4	Rating instrument	Specification of rating instrument, size, range, appearance, use, interface	Interface design
5	Evaluator type & characteristics	Evaluator type/characteristics specification	Recruitment
6	Evaluator recruitment, training	Evaluator recruitment and training protocol	Recruitment

The 11 design steps and outputs from them

Step	Description	Outputs from Phase I (Design) that feed into implementation	Process as part of which implemented
7	Conditions during experiment	Controlled conditions protocol	Response collection
8	Quality assurance	a. QA protocol i (monitoring, checks)	Response collection
		b. QA protocol ii (piloting, testing)	N/A (see Execution)
9	Analysis	Specifications for response normalisation	Response normalisation
		Specifications for aggregation and analysis	Analysis
10	Impact assessment	N/A (completed in Phase I)	N/A
11	Ethical review	N/A (completed in Phase I)	N/A

See corresponding sections in Unit 4.

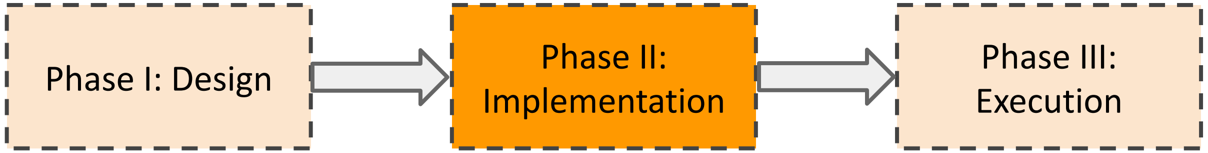
Implementing the component processes

- Will now go through each of the five core processes and two offline processes which have an implementation aspect, discussing the implementation task(s) involved:
    - Output sampling
    - Recruitment
    - Output normalisation
    - Evaluation interface creation
    - Response collection
    - Response normalisation
    - Analysis

contributory (can be offline)

core

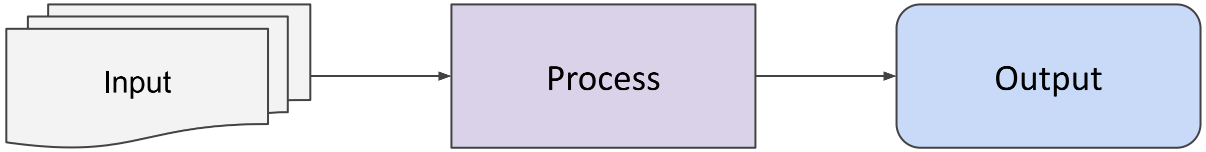
Implementation: an engineering exercise



- Recall Phases I–III: at this stage, we already have the design, now we implement it, ready for execution later.
- Overall aim in implementation phase is to maximise automation to reduce sources of error.

Implementation: an engineering exercise

- Each component process takes data (usually read in from files) as input, performs operations with it in some way, then outputs some other data (usually written to files).



- We will describe each process in turn, in terms of its black-box functionality, what it uses as input, and what it produces as output.
- Note that a given process is not necessarily implemented as a single function or module, but may be implemented as multiple functions/modules.
- The output from each component process becomes the input for the next process.

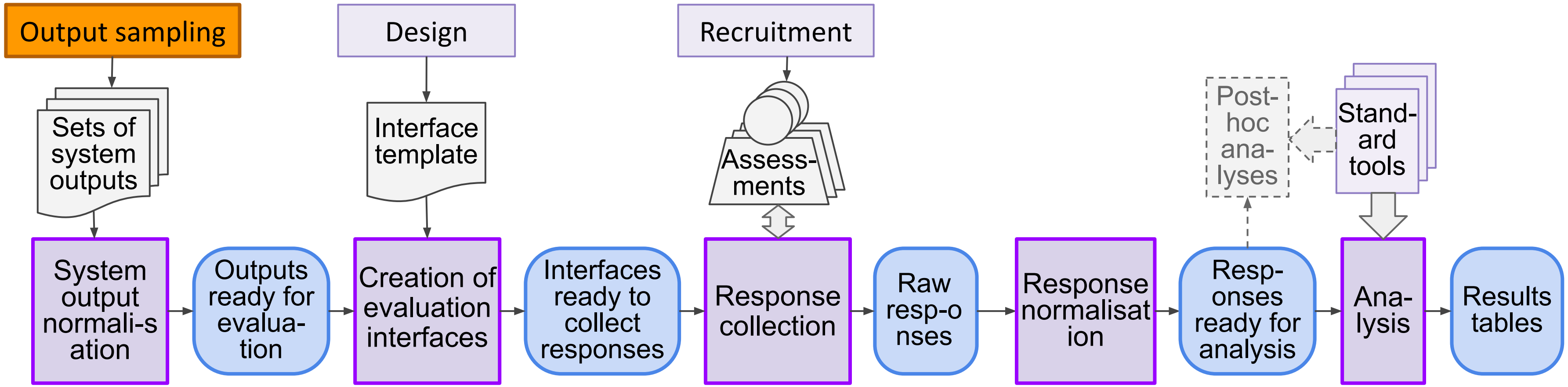
Implementation: an engineering exercise

- We will take a function view of processes to be implemented.
- Process functions usually have data files as inputs and outputs.
- They also have function parameters that were set during design.
- We will distinguish these visually as follows:
  - `Data file`
  - `Function parameters`
  - `Other resources`

Overview

Unit 6: Experiment Implementation

Component process diagram for reference



Implementation of output sampling process

- **Inputs and parameters**: Systems to be evaluated, test set inputs, system outputs for test set inputs where available, target `number of evaluation items`, `number of evaluators`, `type of assignment`, `sampling type`
- **Outputs**: Complete set of system outputs, with identifiers for system and input from which generated; assignment of system outputs to evaluators.
- **Functionality**:
  - Randomly samples a subset of test set inputs of size = `number of evaluation items`, in accordance with `sampling type` specification.
  - Pairs test set inputs with corresponding outputs obtained from all systems to be evaluated.
  - Where system outputs for test set inputs are not available for a given system to be evaluated, generates those.
  - Determines an assignment of system outputs to evaluators according to `type of assignment` and `number of evaluators`.

Implementation of output sampling process

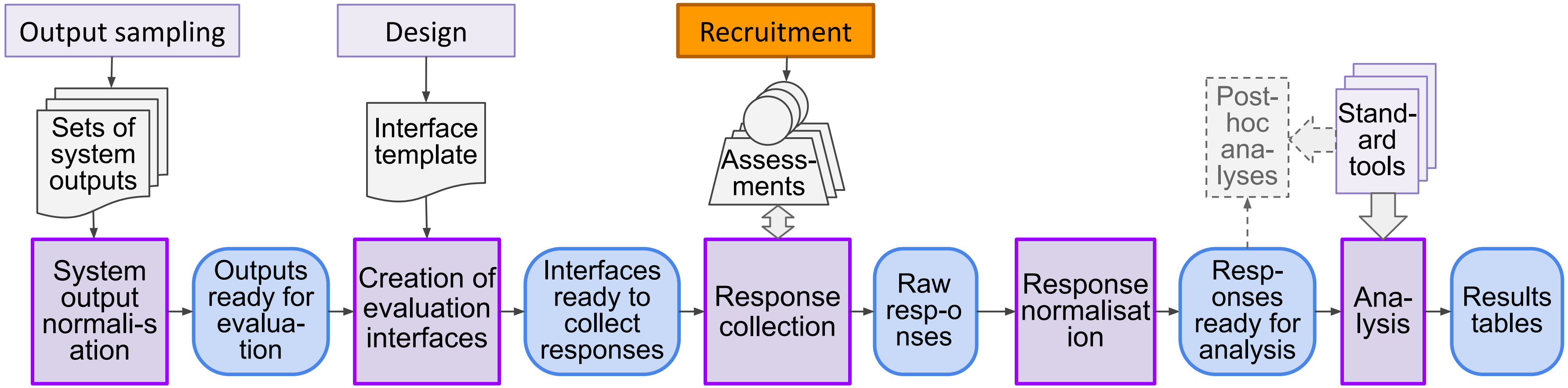
- **Notes on implementation**:
    - Design steps (outputs) based on: Step 3a (Specification for output sampling) (see Table on Slides 7/8).
    - Normally, sampling is on basis of input properties (but outputs is also conceivable).
    - Number of evaluation items per system and number of evaluators per system may need to be adjusted e.g. if there are two many null or damaged outputs from a system.
    - Output sampling process must be done fully automatically.
    - Standard functions available in Python libraries.
    - HEDS question relating to output sampling code can now be completed:

      3.1.3.3: Where can other researchers find details of the sampling code used?
      Link to script used (or another way of identifying the script).

Overview

Unit 6: Experiment Implementation

Component process diagram for reference



Implementation of recruitment process

- **Inputs**: Participant information sheet, informed consent form, participant screening questionnaire, method for approaching participants, e.g., mailing list.
- **Outputs**: List of participants who have been fully trained.
- **Functionality**:
  - If this is a manual process, then it should be a detailed written series of steps required to recruit and train the participants.
  - This process could be automated, e.g., using the Mechanical Turk API, but this is outside the scope of this tutorial.
- **Notes on implementation**:
  - Design steps (outputs) based on: Step 5 (Evaluator type/characteristics specification); Step 6 (Recruitment and training protocol).

Implementation of recruitment process

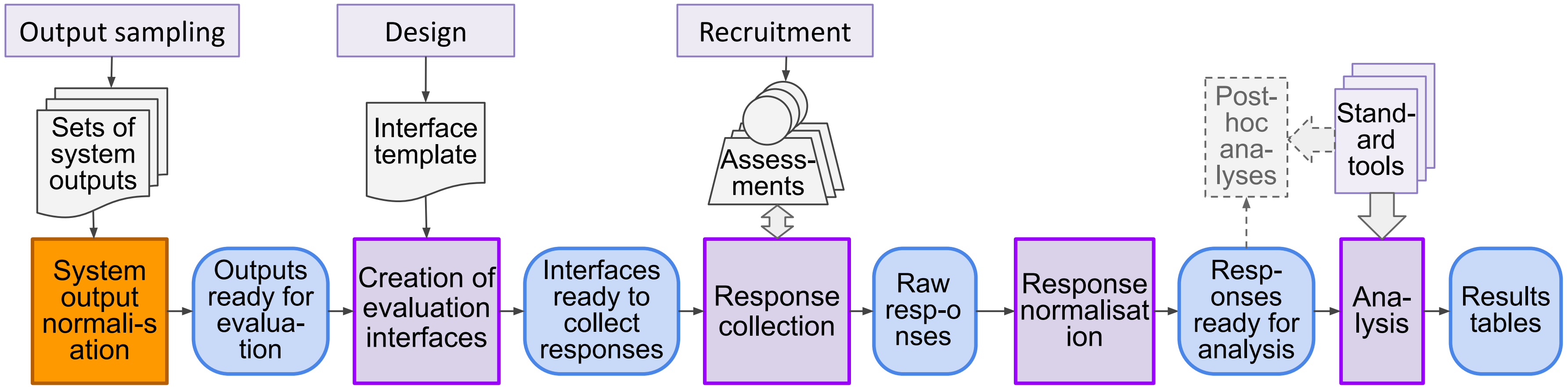
Example:

- Advertise the experiment on departmental mailing list: abc@ouruni.edu.
    - Provide potential participants with a participant information sheet, explaining what they would need to do.
    - Obtain informed consent using an informed consent form (Google Form in this example).
    - Screen participants using the participant screening questionnaire (Google Form in this example; self-reporting expertise and fluency).
- Hold an online training session on Zoom where participants are given time to read the instructions, then perform practice work, and ask any questions.
    - Note: This should be defined in more detail, similar to a lesson plan.
- Securely store the list of participants who have been fully trained.

Overview

Unit 6: Experiment Implementation

Component process diagram for reference



Implementation of system output normalisation process

- **Inputs and parameters:** Set of system outputs, standard text format.
- **Outputs:** Outputs ready for evaluation.
- **Functionality:**
  - Normalise the set of system outputs such that they all adhere to the standard text format (syntax, locale, document structure, etc.).
  - Write a script based on the experiment design that can postprocess the output from each system as required.
  - Store the Outputs ready for evaluation to file.

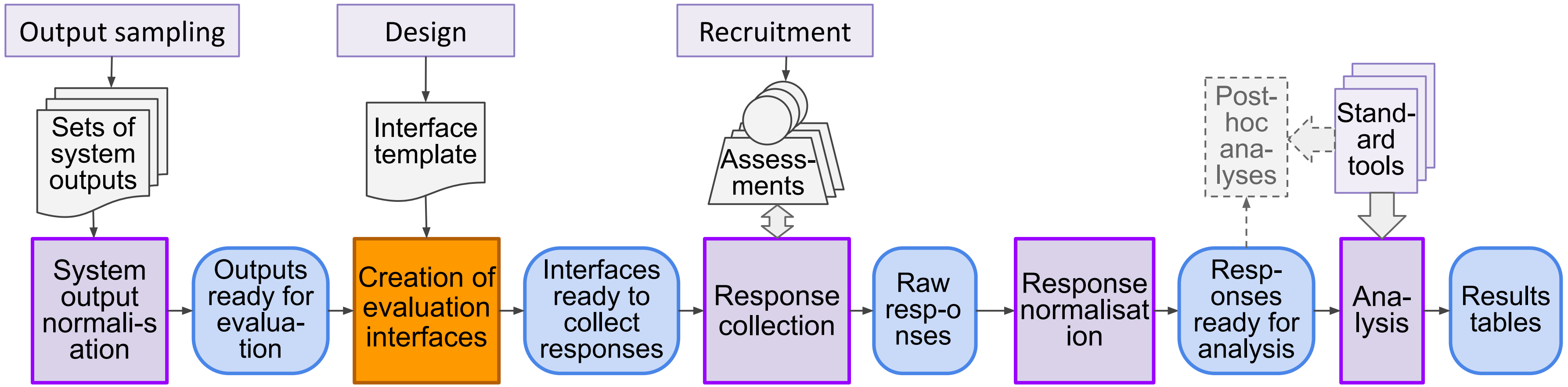
Implementation of system output normalisation process

- **Notes on system output normalisation**:
    - Design steps/outputs based on: 3b/Specifications for output normalisation.
    - Look through system outputs manually to identify any unexpected characters. Regular expressions can additionally be used to check for unexpected characters or sequences.
- **Some examples:**
    - Ensure consistent capitalisation.
    - Ensure consistent syntax and spacing, e.g., the number of spaces following a full stop.
    - Remove any special tags from the model output, e.g., `<START>` and `<END>` tokens.
    - Replace any HTML or other special characters, e.g., `&amp;`

Overview

Unit 6: Experiment Implementation

Component process diagram for reference



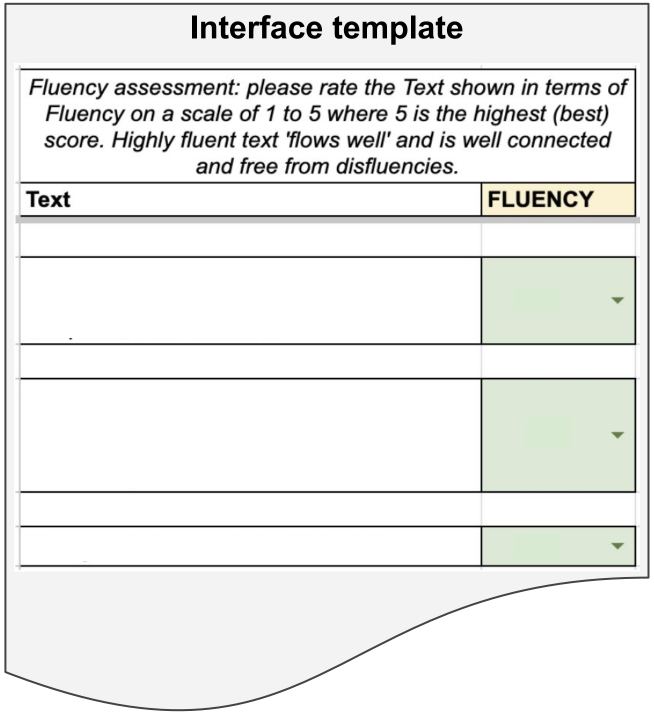
Implementation of evaluation interface creation process

- **Inputs and parameters:** Interface template, outputs ready for evaluation, assignment of system outputs to evaluators
- **Outputs:** Interfaces ready to collect responses.
- **Functionality:**
  - Populate the interface template with the outputs read for evaluation.
  - Assign each generated interface to a real participant, creating interfaces ready to collect responses from the more abstract assignment of system outputs to evaluators.
- **Notes on implementation**.: Will depend on type of interface, e.g.,
  - Spreadsheets.
  - Forms, e.g., Google Forms (automation is self-contained)
  - Web servers.

Implementation of evaluation interface creation process

Example – spreadsheet as interface:

- Implementation procedure:
  - Create a template that is protected except for cells participants must complete.
  - Add validation rules.
  - Create a script that creates from the interface template a spreadsheet for for each participant, containing the appropriate evaluation items.
- Result:
  - A spreadsheet for each participant, containing the appropriate evaluation items.



Interface template

Fluency assessment: please rate the Text shown in terms of Fluency on a scale of 1 to 5 where 5 is the highest (best) score. Highly fluent text 'flows well' and is well connected and free from disfluencies.

Text	FLUENCY

Implementation of evaluation interface creation process

Example – Crowdwork platform:

- Implementation procedure:
  - Create an HTML template that will be populated based on values from corresponding column names in the evaluation items CSV.
  - Add validation rules to the HTML template using Javascript.
  - Setup the project on the platform.
  - Upload the CSV of evaluation items to the platform.
- Result:
  - Mechanical Turk-like Project and configuration setting that can be run on the platform.

Implementation of evaluation interface creation process

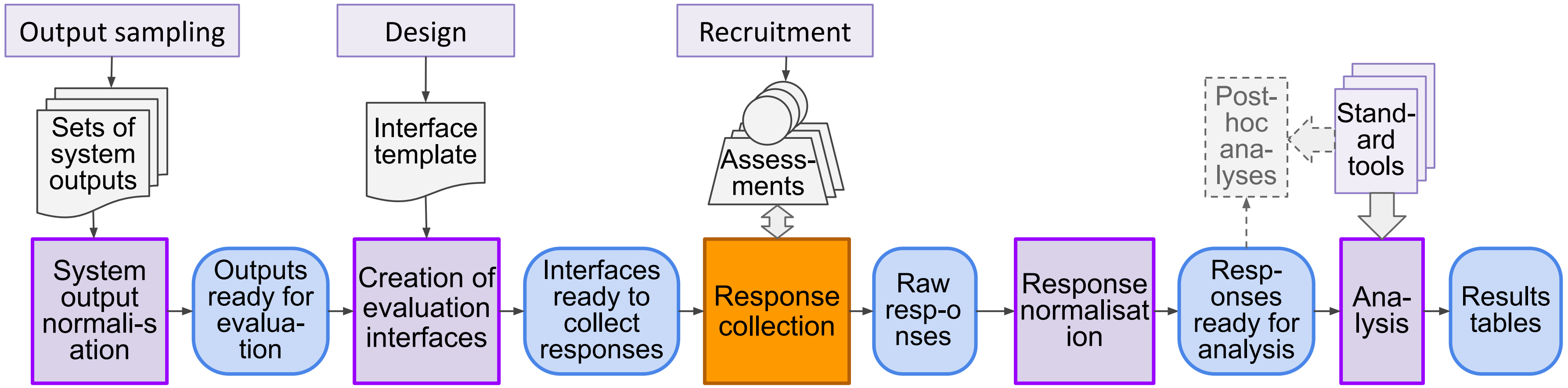
Example – using a web server to present evaluation interfaces:

- Implementation procedure:
  - **Custom web server**: will depend on the design requirements and is outside the scope of this tutorial.
  - **Complex survey platform**: Such as Qualtrics or Potato
- Some crowd platforms support integrations to web servers, or provide their own interface design tools:
  - Prolific supports integration with dozens of interface platforms.
  - Mechanical Turk has its own HTML template engine.
- General purpose, open source, interface platforms have been developed to emulate and improve upon the functionality offered by platforms such as Mechanical Turk (Watson & Gkatzia, 2024).

Overview

Unit 6: Experiment Implementation

Component process diagram for reference



Implementation of response collection process

- **Inputs and parameters:** Interfaces ready to collect responses, list of participants
- **Outputs:** Raw responses.
- **Functionality:**
    - Serve the fully instantiated evaluation interfaces to participants and store their responses in a database or to file as Raw responses.

Implementation of response collection process

- **Notes on implementation**:
  - If using a crowdwork platform, this process is provided by the platform and may take a slightly different form, e.g., you might not have a list of participants in advance.
  - For spreadsheet interfaces, you can create a mail merge using an automatically generated spreadsheet where each row contains the participants details (name, email), and a URL for the spreadsheet (hosted in the cloud) they need to complete. Then, create a script that collates responses from each spreadsheet that is in the cloud.

Implementation of response collection process

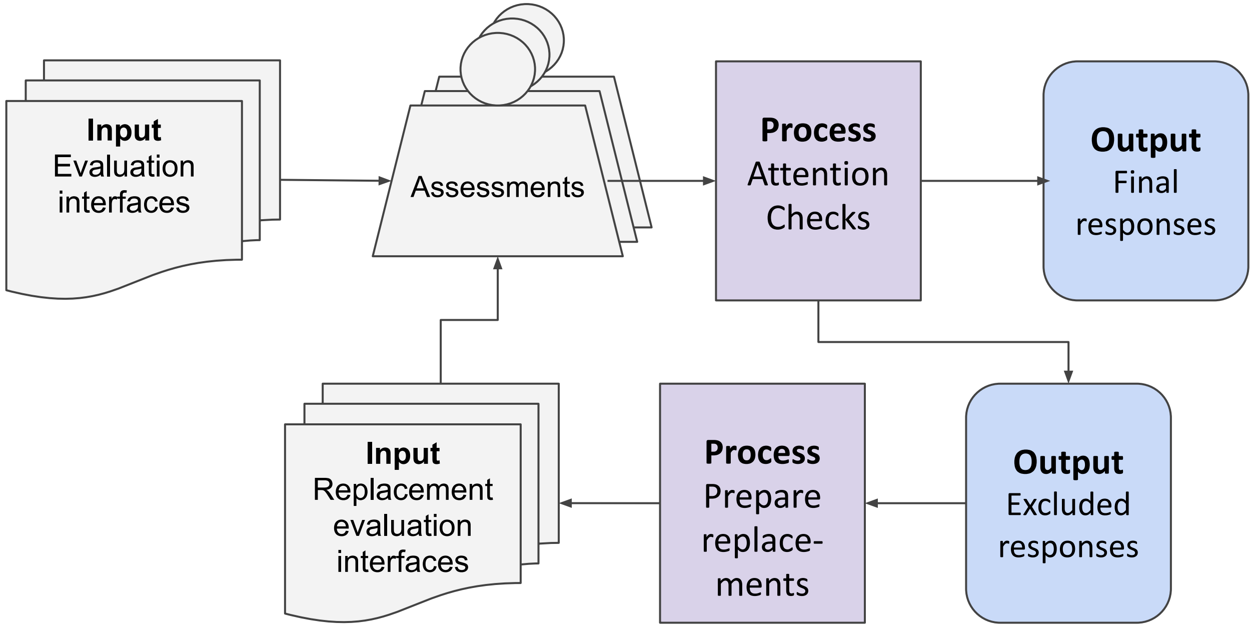
Attention checks and excluding responses:

- We sometimes need to exclude the responses of a participants because the participant has failed attention checks.
- The exclusion criteria must be defined in the design.
- Code can be written in advance that applies attention checks and generates new interface files that can be used to obtain replacement responses.
- Retain snapshots of the responses (original and replacements) at each stage and iteration.

Diagram of response exclusion loop

Response exclusion loop:

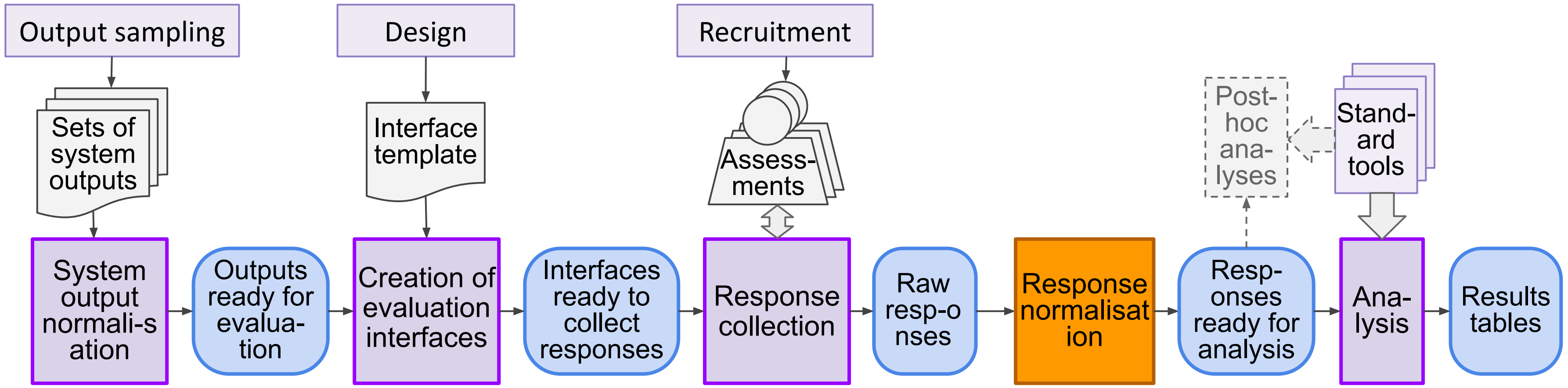
- The below diagram shows a loop of collecting replacement responses because of failed attention checks, before finally applying any outlier exclusion (Thomson & Belz, 2024).



Overview

Unit 6: Experiment Implementation

Component process diagram for reference



Implementation of response normalisation process

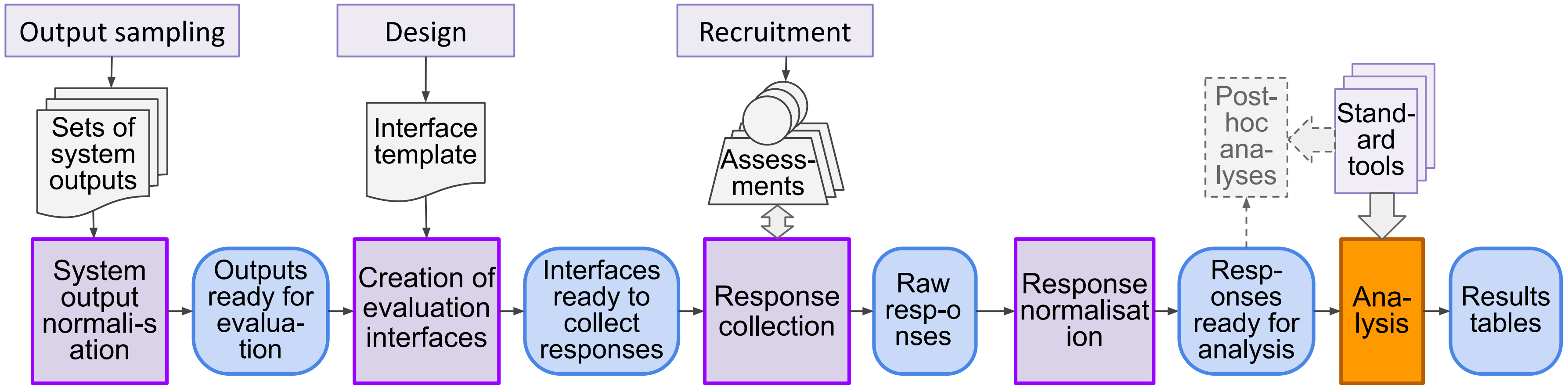
Postprocessing responses:

- **Inputs and parameters:** Raw Responses.
- **Outputs:** Responses ready for analysis.
- **Functionality:** Process the Raw Responses:
    - Check that all responses are present and within the allowed range of the rating instrument.
    - Check there are no untoward patterns, e.g., answering 3 to every question.
    - Convert the data to a normalized format (1 row per response, identifiers, system labels, etc., anonymised annotator names).
    - Save the resultant data to file as Responses ready for analysis.
- **Notes on implementation:**
    - May need to collect replacement responses in the event of invalid responses.

Overview

Unit 6: Experiment Implementation

Component process diagram for reference



Implementation of analysis process

- **Inputs and parameters:** Responses ready for analysis, pre registered analysis methods
- **Outputs:**
  - Data files.
  - Tables / figures.
- **Functionality:** A script which applies the pre registered analysis methods to the responses ready for analysis. These are then converted into tables / figures for inclusion in a paper or other report, as well as data files., i.e., a spreadsheets, containing the underlying numerical values.

Implementation of analysis process

- **Notes on implementation:**
  - Since the form of the responses and the research questions are known in advance of running the experiment, we can write a code script for the analysis in advance, and even test it with synthetic data.
  - See Unit 5 for more details on analysis.

Implementation of analysis process

Implementing the method:

- The primary analysis method(s) are specified in the experiment design. E.g., for an intrinsic, subjective, and absolute evaluation, we may have planned to:
  - Test the distribution of our responses using the Shapiro–Wilk test, then:
    - **IF** (normally distributed): Analysis of variance (ANOVA)
    - ***ELSE***: Kruskal–Wallis test
  - Determine inter-annotator agreement:
    - Krippendorff's alpha
- Because the format of our responses (raw and aggregated) is known in advance, we can implement (in code) and test this analysis before we run the experiment.
  - See Thomson & Belz. 2024 for an example.
- See Unit 5 for more information on appropriate analyses.

Implementation of analysis process

Generating results tables:

- Automating the generation of tables reduces human error.
- Because our statistical analysis has been encoded, we can easily append the generation of results tables to the code.
    - Using functions such as `pandas.DataFrame.to_latex`.
    - With your own custom scripts based on the types of tables you need.
- It is good practice to include a copy of each table as data in supplementary material as well, e.g., as a CSV file.

Overview

Unit 6: Experiment Implementation

Pipelining

- It is usually possible to fully automate within each component process.
- Some manual work may be required in order to pipeline the component processes. This should be kept to a minimum.
- For example, the research may need to run a separate script in order to execute each component process. The exact commands used should be recorded (copy and paste it from the shell history, or copy it from the implementation documentation in the first place).
- If using a Jupyter notebook for pipelining, the notebook should run without errors on a fresh kernel when the “run all” option is selected.

Implementation documentation

- If you are going to have to do something, write it down first!
- Create a list of all steps that you will need to complete in order to execute the experiment.
  - It should be possible for any researcher to follow these instructions.
  - If code needs to be run, write down the commands. Ideally we would combine several commands together using a shell scripts to form pipelines.
  - Any non-automated component processes should be fully documented here, e.g., the component process of delivering each interface to the appropriate participant.

Updating the design

- The process of implementing the design is likely to raise issues that result in adjustments to the design.
- The design should be updated, including making changes to the HEDS, either prior to or at the time of implementing any changes.
- We should never have undocumented, ad hoc changes.

Overview

Unit 6: Experiment Implementation

Good coding practices

- All software may contain bugs. We can expect once or two errors per hundred lines of code, in software that has not gone through rigorous testing (McConnel, 2004).
- It is not practical to submit our experiment code to the same level of software testing that might be seen in industry applications.
- However, there are still things we can (and should) do.
- There are resources available:
    - The Good Research Handbook (Mineault & Community, 2021), is an excellent resource for researchers, especially those using Python.

Setting up the code project

- Use tools such as Cookie Cutter (cookiecutter.io), e.g., default file and folder layouts for your human evaluation projects.
- Style guides & linters, e.g.,
    - pylint
    - Black
    - ruff
- Know when (and when not) to use Jupyter notebooks.

Writing clean code

- Learn to identify and use pure functions
- Avoid side effects
- Split IO and computation
- Decrease the amount of nesting
- Improving legibility

Testing

- It is not practical to write full test suites for all experiment code. However, we may want to consider doing so for functions or libraries that we commonly use across different experiments.
- Simple checks, such as `assert` statements in Python, can be used. E.g.,
  - `assert len(items) == OUTPUTS_PER_SYS * N_SYSTEMS`
- We can also write simple tests for things such as participant-item combinatorics. If, for example, we know the below, we can check it is reflected in our response file:
  - `PARTICIPANTS_PER_ITEM = 3`
  - `ITEMS_PER_PARTICIPANT = 36`

Code documentation

- Write meaningful error messages.
- Use in-line comments.
- Use Docstrings
- Generate documentation, e.g., on ReadTheDocs.

Code review

- Check code yourself at least a day after it was written, rather than “write and forget”.
- Have someone else check your experiment code.
- A code review process can be set up in your lab.

Overview

Unit 6: Experiment Implementation

Unit summary and pointers to other units

- Unit 6 looked at the **experiment implementation**, based on the **experiment design** from Unit 4 and the planned **analysis** (Unit 5)
- This was approached as an engineering exercise where we implement the design and then ensure its quality through code testing and review.
- Issue related to the implementation of each **component process** from Unit 4 were discussed:
  - Output sampling
  - Recruitment
  - Output normalisation
  - Evaluation interface creation
  - Response collection
  - Response normalisation
  - Analysis

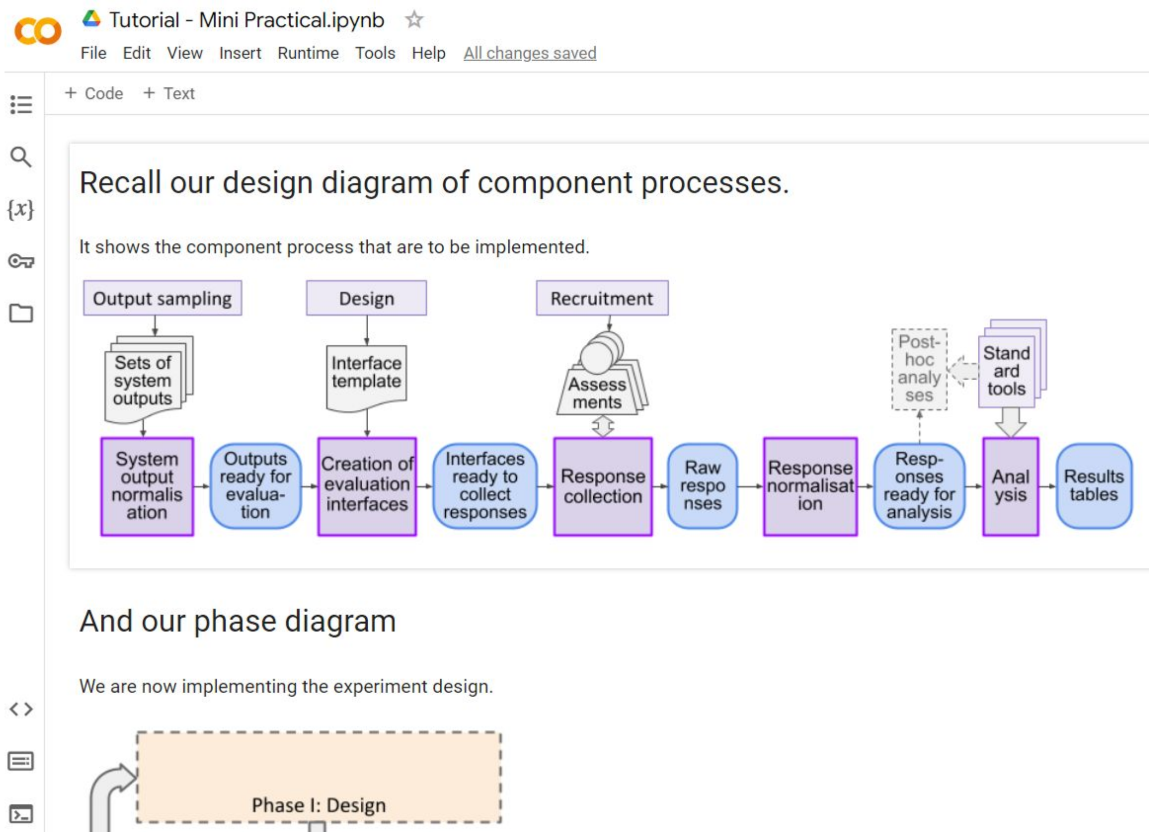
Unit summary and pointers to other units (cont.)

- Also discussed was the pipelining of these component processes (where possible)
- Manually processes were permitted in the architecture, although these were carefully written out in advance so that any researcher could follow them.

Overview

Unit 6: Experiment Implementation

Practical - High-level Colab walkthrough



Tutorial - Mini Practical.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

Recall our design diagram of component processes.

It shows the component process that are to be implemented.

Output sampling
Design
Recruitment
Sets of system outputs
Interface template
Assess ments
Post-hoc analy ses
Stand ard tools
System output normalis ation
Outputs ready for evalua-tion
Creation of evaluation interfaces
Interfaces ready to collect responses
Response collection
Raw respo nses
Response normalisat ion
Resp-onses ready for analysis
Anal ysis
Results tables

And our phase diagram

We are now implementing the experiment design.

Overview

Unit 6: Experiment Implementation

References

- Essential reading:

The Good Research Code Handbook.
Mineault, P. (2021). The Good Research Code Handbook Community.

(Mostly) Automatic Experiment Execution for Human Evaluations of NLP Systems.
Craig Thomson and Anya Belz. 2024. In *Proceedings of the 17th International Natural Language Generation Conference*, pages 272–279, Tokyo, Japan. Association for Computational Linguistics.