

LISTER: Life Science Experiment Metadata Parser

This repository contains a set of files to parse SOP from lab experiments.

Motivation

As research group usually has its own set of SOP to conduct experiments, a tool to extract metadata from an editable document (e.g., DOCX) would be handy. The metadata is helpful in documenting the research and hence improves the reproducibility of the conducted research. To enable the metadata extraction, the experiment documentation should follow some annotation rules (described later below).

Repository structure

- The base directory contains the metadata extraction script.
- *input* directory contains the docx experiment documentation examples for the extraction.
- *output* directory contains extracted steps order – key – value in both JSON and XLSX format.

List of supported annotations

The parser extracts:

Extracted Items	Description	Representation	Example	Extracted order,key,value
Section	The section name	<section section name>	<section Structure Preparation>	<ul style="list-style-type: none"> "-", section, Structure Preparation
Order	The <i>order</i> of the steps, based on the order of the paragraph in the docx SOP document	-	-	-
Key	The <i>key</i> for the metadata, based on the value represented in the curly bracket after the pipe character {value key}.	{value key}	{sequence alignment stage}	<ul style="list-style-type: none"> <order>, stage, sequence alignment
Comment	<i>Comments</i> are allowed within the key, represented within regular brackets after the pipe symbol. Comment can be placed both/either before and/or after key and/or value. TBD: How to serialize comments in the metadata parser output.	{value (comment) key} or {value (comment) key} or {value (comment) (comment) key}	{receptor residue (minimization) target}	<ul style="list-style-type: none"> <order> target, receptor residue

Extracted Items	Description	Representation	Example	Extracted order,key,value
Value	The <i>value</i> of the metadata is based on the first value represented in the curly bracket before the pipe character {value key}. Example: with 'sequence alignment' as the value.	{value key}	{sequence alignment stage}	<ul style="list-style-type: none">• <order>, stage, sequence alignment
Control flow: for each	Extract multiple key value pairs related to for each iterations	< for each iterated value>	< for each generated pose>	<ul style="list-style-type: none">• <order>, step type, iteration• <order>, flow type, for each• <order>, flow parameter, generated pose

Extracted Items	Description	Representation	Example	Extracted order,key,value
Control flow: while	Extract multiple key value pairs related to while iteration	< while key logical operator value> ... <iteration operation magnitude>	< while pH lte 7> ... <iterate + 1>	<ul style="list-style-type: none">• <order>, step type, iteration• <order>, flow type, while• <order>, flow parameter, pH• <order>, flow logical parameter, lte• <order>, flow compared value, 7• <order>• flow type, iterate (after while)• flow operation, +,• <order>, flow magnitude, 1

Extracted Items	Description	Representation	Example	Extracted order,key,value
Control flow: if	Extract multiple key value pairs related to if iteration	<`if` key logical operator value>	< if pH lte 7>	<ul style="list-style-type: none">• <order>, step type, <i>conditional</i>• <order>, flow type, <i>if</i>• <order>, flow parameter, pH• <order>, flow logical parameter, lte• <order>, flow compared value, 7

Extracted Items	Description	Representation	Example	Extracted order,key,value
Control flow: else if	Extract multiple key value pairs related to else if iteration	< else if key logical operator value>	< else if pH between [8-12]>	<ul style="list-style-type: none">• <order>, step type, conditional• <order>, flow type, else if• <order>, flow parameter, pH• <order>, flow logical parameter, between• <order>, flow range, [8-12]• <order>,start iteration value,8• <order>,end iteration value,12
Control flow: else	Extract multiple key value pairs related to else iteration	< else >		<ul style="list-style-type: none">• <order>, step type, conditional• <order>, flow type, else

Extracted Items	Description	Representation	Example	Extracted order,key,value
Control flow: for	Extract multiple key value pairs related to for iteration	< for key [range] iteration operation magnitude>	< for pH [1-7] + 1>	<ul style="list-style-type: none">• <order>, step type, iteration• <order>, flow type, for• <order>, flow parameter, pH• <order>, flow logical parameter, lte• <order>, flow flow range, [1-7]• <order>,start iteration value,1• <order>,end iteration value,7• <order>, flow operation, +,• <order>, flow magnitude, 1

The overall example of the SOP document is available in the *input/sop2.docx* file. The color formatting in the *sop2.docx* does not play any role in the order/key/value extraction.

Supported operators

Logical operator

Logical operator is used to decide whether a particular condition is met during iteration/conditional block. It is available for `while`, `if` and `else if` control flow. The following logical operators are supported:

- `e` : equal
- `ne` : not equal
- `lt` : less than
- `lte` : less than equal
- `gt` : greater than
- `gte` : greater than equal
- `between` : between

Iteration operator

Iteration operator is used to change the value of compared variable during a loop. It is available for `while` and `for`. The following iteration operators are supported:

- `+` : iteration using addition
- `-` : iteration using subtraction
- `%` : iteration using modulo
- `*` : iteration using multiplication
- `/` : iteration using division

Document validation

LISTER checks the following problems upon parsing, and report accordingly:

- Orphaned brackets and indicates which line the error is located.
- Mismatched data types for conditionals and iterations.
- Mismatched argument numbers for conditionals and iterations.
- Invalid control flows.

Constraints and recommendations

Constraints

- Subprocess/substep is not currently supported to simplify data storage process (or let me know if this is really necessary, providing suggestion on how to store/serialize it would be great). Any subprocess will be stored simply as a next step from its parent process.
- Comments are currently not yet extracted in the parser's output but it is already parsed in the background - it is mainly intended to provide more detail about a particular key/value for SOP adopters.

Recommendations

- Avoid the use of reference without explicit KV-pair (avoid e.g., "*Repeat step 1 with similar parameters*"), as this will make the metadata for that particular implicit step unextracted.
- To minimize confusion regarding units of measurement (e.g., `fs` vs `ps`), please explicitly state the units within the value portion of the KV-pair, e.g., `{0.01 ps|gamma_1n}`.

Running the parser

Using a *.docx file as an input

Requirement: a *.docx file adhering to annotation rules.

1. Create SOP according to the above annotation rules, save it into a *.docx file.
2. Ensure that the code under `PARSING FROM ELABFTW CONTENT` block is commented.
3. Uncomment the code block under `# PARSING FROM DOCX DOCUMENT`.
4. Change the output directory/filename (last line).
5. Change the input directory/file name in the python script.
6. Run the script.

Using an eLabFTW experiment entry via API

Requirement: an eLabFTW experiment entry adhering to annotation rules, the experiment number from that experiment and eLabFTW API Token from your user account.

1. Create SOP according to the above annotation rules, save it into an experiment.
2. Ensure that the code under `# PARSING FROM DPCX DOCUMENT` block is commented.
3. Uncomment the code block under `# PARSING FROM ELABFTW CONTENT` .
4. Change the output directory/filename (last line).
5. Change the input directory/file name in the python script.
6. Run the script.

Open for discussion

1. Comments serialization, as well as representing and structuring comments in the output file.
2. **The necessity of supporting substeps parsing and serialization.**
3. Support for markdown:
 1. This will require redesign on how the text should be annotated, as the bracketing method will break or at least requires a lot of metacharacters.
 2. On the plus side, MD is supported by eLabFTW, also the changes can be tracked on version control system.
4. Maintaining information in the form of images, tables, figures etc.
5. Integration with eLabFTW.
6. Whether integration with SWATE and ARC is feasible.
7. Creating ontology terms from collected SOPs, and linking the keys with the ontologies.
8. How step number should be counted, e.g., should it be restarted from 1 after a new section.
9. **How can we accommodate the paragraph with similar key having several values? (mainly for me).**
10. How to map enumerated values of one key into corresponding values of a related key?

Further plans

1. Implement TODOs in this readme, and bug fixes.
2. Gather feedback from LISTER users.
3. Consult CAi and Biochemistry1 for LISTER's implementability on other labs.

4. Align the used keys with terms from an ontology, or if the term does not exist, create a new term by extending an ontology or creating a term within a new ontology.