DOCX Standard Operating Procedure (SOP) Parser for SFB1208 Experiments

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

Why?

As research lab usually has their 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 SOP should follow some annotation rules (described later below).

How is this repo structured?

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

What is extracted from the SOP, and how is it represented in the docx document?

The parser extracts:

Extracted Items	**Description**	**Representation**	**Example**	**Extracted order,key,value**
Section	The section name	*<section|section name>*	*<section|Structure Preparation>*	• "-", *section*, *Structure Preparation*
Order	The *order* of the steps, based on the order of the paragraph in the docx SOP document	-	-	-
Key	The *key* for the metadata, based on the value represented in the curly bracket after the pipe character {value|key}.	{value|*key*}	{sequence alignment|*stage*}	• <order>, *stage*, sequence alignment
Comment	*Comments* are allowed within the key, represented within regular brackets after the pipe symbol. **TBD**: How to serialize comments in the metadata parser output.	{value|(*comment*) key}	{receptor residue|(*minimization*) target}	• <order> target, receptor residue
Value	The *value* 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}	• <order>, stage, *sequence alignment*
Control flow: *for each*	Extract multiple key value pairs related to *for each* iterations	*<flow type*|iterated value>	*<for each|generated pose>*	• <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	<flow type|key|logical operator|value> ... <iteration operation|magnitude>	<while|pH|lte|7> ... <+|1>	• <order>, step type, *iteration* • <order>, flow type, *while* • <order>, flow parameter, pH • <order>, flow logical parameter, lte • <order>, flow compared value, 7 • <order>, 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>	• <order>, step type, *conditional* • <order>, flow type, *if* • <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]>	• <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>		• <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>	• <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 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

How the parser should be run?

1. Create SOP according to the above annotation rules.
2. Change the input directory/file name in the python script (2nd last line).
3. Change the output directory/filename (last line).
4. Run the script.

What are the further plans?

1. Fixes for while control flow, and logical operators in general control flow.
2. Consult CAi and Biochemistry1 for its implementability on other labs.
3. 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.