OCEL Feature Extraction Documentation

November 2021 - ??? 2022

Contents

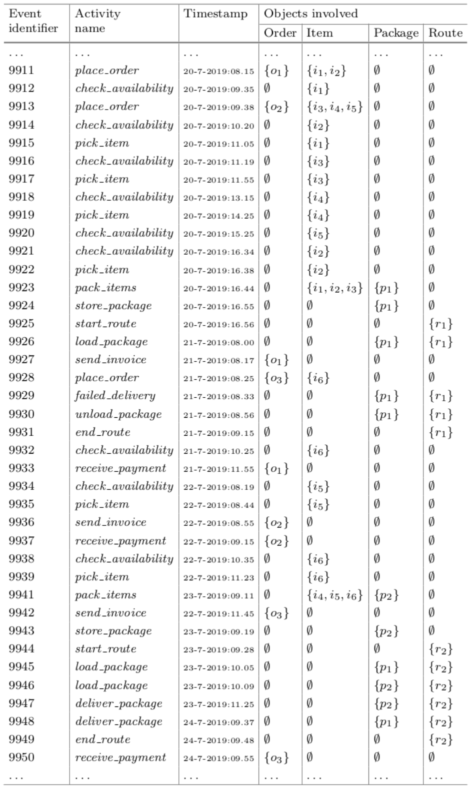
1 Document Format

The structure section will go over all the structures that are created in order for many of the feature extraction methods to function. Each structure will have information on input, context behind the structure and a sample image to demonstrate the output on a basic example.

The Feature sections go over feature extraction methods explored throughout the thesis on the topic. All features are aimed at being very general, applicable to any input ocel log. The descriptions will include the input, structures used, output, context and, if required, a very small basic visualization based on the examples that are found in the structures section.)

2 Example OCEL

The following log was taken from ... and will be used for the majority of examples in this document. Of course this log is very basic and will therefore not cover all edge cases that can occur in ocel logs. However, using the log, I hope to show the intuition of all structures and feature extraction methods.



Event identifier	Activity name	Timestamp	Objects involved			
			Order	Item	Package	Route
...	...	...	...	...	...	...
9911	*place_order*	20-7-2019:08.15	$\{o_1\}$	$\{i_1, i_2\}$	$\emptyset$	$\emptyset$
9912	*check_availability*	20-7-2019:09.35	$\emptyset$	$\{i_1\}$	$\emptyset$	$\emptyset$
9913	*place_order*	20-7-2019:09.38	$\{o_2\}$	$\{i_3, i_4, i_5\}$	$\emptyset$	$\emptyset$
9914	*check_availability*	20-7-2019:10.20	$\emptyset$	$\{i_2\}$	$\emptyset$	$\emptyset$
9915	*pick_item*	20-7-2019:11.05	$\emptyset$	$\{i_1\}$	$\emptyset$	$\emptyset$
9916	*check_availability*	20-7-2019:11.19	$\emptyset$	$\{i_3\}$	$\emptyset$	$\emptyset$
9917	*pick_item*	20-7-2019:11.55	$\emptyset$	$\{i_3\}$	$\emptyset$	$\emptyset$
9918	*check_availability*	20-7-2019:13.15	$\emptyset$	$\{i_4\}$	$\emptyset$	$\emptyset$
9919	*pick_item*	20-7-2019:14.25	$\emptyset$	$\{i_4\}$	$\emptyset$	$\emptyset$
9920	*check_availability*	20-7-2019:15.25	$\emptyset$	$\{i_5\}$	$\emptyset$	$\emptyset$
9921	*check_availability*	20-7-2019:16.34	$\emptyset$	$\{i_2\}$	$\emptyset$	$\emptyset$
9922	*pick_item*	20-7-2019:16.38	$\emptyset$	$\{i_2\}$	$\emptyset$	$\emptyset$
9923	*pack_items*	20-7-2019:16.44	$\emptyset$	$\{i_1, i_2, i_3\}$	$\{p_1\}$	$\emptyset$
9924	*store_package*	20-7-2019:16.55	$\emptyset$	$\emptyset$	$\{p_1\}$	$\emptyset$
9925	*start_route*	20-7-2019:16.56	$\emptyset$	$\emptyset$	$\emptyset$	$\{r_1\}$
9926	*load_package*	21-7-2019:08.00	$\emptyset$	$\emptyset$	$\{p_1\}$	$\{r_1\}$
9927	*send_invoice*	21-7-2019:08.17	$\{o_1\}$	$\emptyset$	$\emptyset$	$\emptyset$
9928	*place_order*	21-7-2019:08.25	$\{o_3\}$	$\{i_6\}$	$\emptyset$	$\emptyset$
9929	*failed_delivery*	21-7-2019:08.33	$\emptyset$	$\emptyset$	$\{p_1\}$	$\{r_1\}$
9930	*unload_package*	21-7-2019:08.56	$\emptyset$	$\emptyset$	$\{p_1\}$	$\{r_1\}$
9931	*end_route*	21-7-2019:09.15	$\emptyset$	$\emptyset$	$\emptyset$	$\{r_1\}$
9932	*check_availability*	21-7-2019:10.25	$\emptyset$	$\{i_6\}$	$\emptyset$	$\emptyset$
9933	*receive_payment*	21-7-2019:11.55	$\{o_1\}$	$\emptyset$	$\emptyset$	$\emptyset$
9934	*check_availability*	22-7-2019:08.19	$\emptyset$	$\{i_5\}$	$\emptyset$	$\emptyset$
9935	*pick_item*	22-7-2019:08.44	$\emptyset$	$\{i_5\}$	$\emptyset$	$\emptyset$
9936	*send_invoice*	22-7-2019:08.55	$\{o_2\}$	$\emptyset$	$\emptyset$	$\emptyset$
9937	*receive_payment*	22-7-2019:09.15	$\{o_2\}$	$\emptyset$	$\emptyset$	$\emptyset$
9938	*check_availability*	22-7-2019:10.35	$\emptyset$	$\{i_6\}$	$\emptyset$	$\emptyset$
9939	*pick_item*	22-7-2019:11.23	$\emptyset$	$\{i_6\}$	$\emptyset$	$\emptyset$
9941	*pack_items*	23-7-2019:09.11	$\emptyset$	$\{i_4, i_5, i_6\}$	$\{p_2\}$	$\emptyset$
9942	*send_invoice*	22-7-2019:11.45	$\{o_3\}$	$\emptyset$	$\emptyset$	$\emptyset$
9943	*store_package*	23-7-2019:09.19	$\emptyset$	$\emptyset$	$\{p_2\}$	$\emptyset$
9944	*start_route*	23-7-2019:09.28	$\emptyset$	$\emptyset$	$\emptyset$	$\{r_2\}$
9945	*load_package*	23-7-2019:10.05	$\emptyset$	$\emptyset$	$\{p_1\}$	$\{r_2\}$
9946	*load_package*	23-7-2019:10.09	$\emptyset$	$\emptyset$	$\{p_2\}$	$\{r_2\}$
9947	*deliver_package*	23-7-2019:11.25	$\emptyset$	$\emptyset$	$\{p_2\}$	$\{r_2\}$
9948	*deliver_package*	24-7-2019:09.37	$\emptyset$	$\emptyset$	$\{p_1\}$	$\{r_2\}$
9949	*end_route*	24-7-2019:09.48	$\emptyset$	$\emptyset$	$\emptyset$	$\{r_2\}$
9950	*receive_payment*	24-7-2019:09.55	$\{o_3\}$	$\emptyset$	$\emptyset$	$\emptyset$
...	...	...	...	...	...	...

Figure 1: Simple example log

3 Structures

3.1 Object-Based

3.1.1 Object Interaction Graph

Description - This *undirected* graph simply places all objects on the graph as *nodes* and assigns *edges* based on whether an object interacted in the same event as the other.

Context - The goal of this graph is to provide a base graph that allows relationships between objects to be explored.

Strengths

1. Very simple
2. Allows for initial relationships between all objects

limitations

1. Very high number of edges
2. An object that interacts a lot with a target object is displayed the same as an object with very few interaction

Visual example

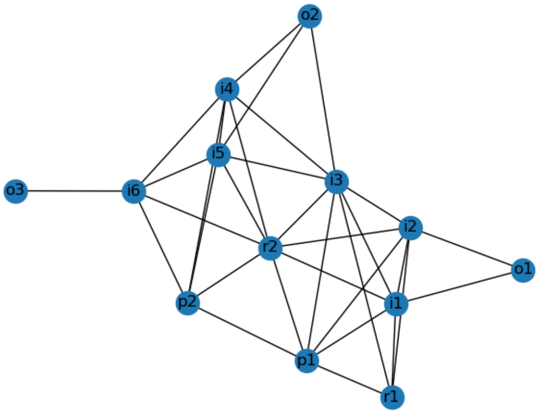


Figure 2: Example of an Object Interaction Graph

3.1.2 Object Descendant Graph

Description - This *directed* graph places all objects on the graph as *nodes* and assigns *edges* based on whether an source object participated in the event where the target object participated in its first event. Note that this also means that objects that were created in the same event are descendants of each other.

Context - The goal of this graph is to take advantage of the time dimension in order to separate objects from each other. This helps greatly reduce the degree of each of the nodes.

Strengths

1. More focussed graph on objects that are related by the time dimension.
2. Direction adds ability to easily segment further (via connectedness) and find independence between objects.
3. Allows for disconnected subgraphs

limitations

1. Ignores object relationships related to frequency of interaction with each other.
2. Can cause many objects to have no edges.
3. General objects (eg. SYSTEM user) will have a very large descendants list.)

Visual example

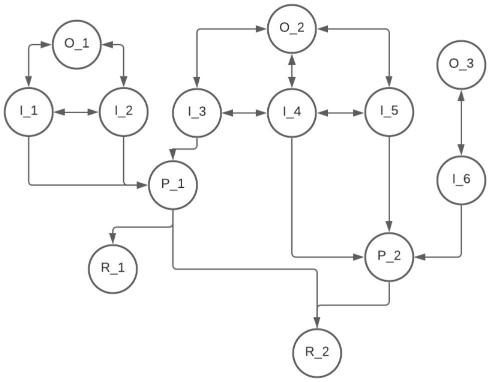


Figure 3: Example of an Object Descendant Graph

3.2 Object Lineage Graph

Description -
Context -
Strengths
Limitations
Visual Example

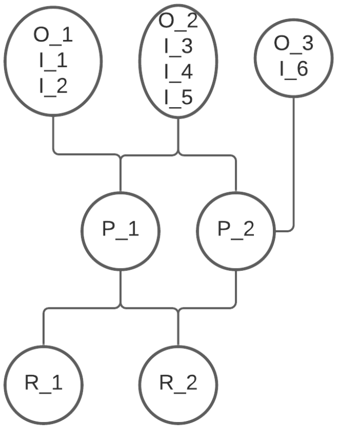


Figure 4: Example of a Lineage Graph

3.3 Event-Based

4 Object Based Features

4.1 Point wise Features

This section includes features that are related to single objects. This means that each object produces its own vector which can be used for further computation.

4.1.1 Activity Existence

Property	**Value**
User Input	None
Structure Input	None
Output	OHE of activity names per object
Function Name	add_activity_existence

Context - The goal of this feature is to get a representation of which activities each object participates in. This gives an overall view of how an object participates does action in a log.

4.1.2 Object Lifetime

Property	Value
User Input	None
Structure Input	None
Output	uint64 seconds an object lived
Function Name	add_object_lifetime

Context - The goal of this feature is to understand how long an object was interacting in the system.

4.1.3 Object Unit Set Ratio

Property	Value
User Input	None
Structure Input	None
Output	float64 [0,1] ratio being a unit set in type
Function Name	add_obj_unit_set_ratio

Context - The goal of this feature is to understand what type of object it is. Whether it operates alone as a type (eg. Orders) or with many other of the same type (eg. items) throughout all events it takes part in.

4.1.4 Average number of other objects in Events

Property	Value
User Input	None
Structure Input	None
Output	float64 avg object interactions per event
Function Name	add_avg_obj_event_interaction

Context - The goal of this feature is to understand how social the object is with any other type of object while executing events.

4.1.5 Unique Neighbor Count

Property	Value
User Input	None
Structure Input	Object Graph
Output	uint64 number of neighboring objects
Function Name	add_unique_neighbor_count

Context - The goal of this feature is to understand with how many objects in total the target object is working with.

Visual example

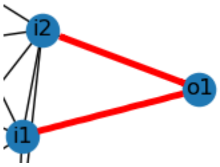


Figure 5: o1 has two unique neighbors. i1 and i2.

4.1.6 Object Type Interaction Count

Property	Value
User Input	Object Type (Default: All)
Structure Input	Object Graph
Output	uint64 number for each input object type
Function Name	TBA

Context - The goal of this feature is to quantize to what type of objects the target object is interacting with in all events.

Visual example

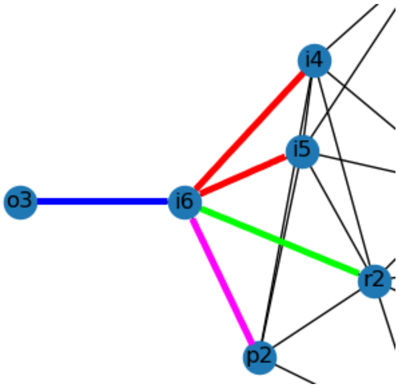


Figure 6: Object i6 has 5 neighbors. There is 1 of order, package, route and 2 are of type item

4.1.7 starting or ending object

Property	Value
User Input	None
Structure Input	Object Lineage Graph
Output	bool for each column
Function Name	TBA

Context - The goal of this feature is to see whether the target object is a root or a leaf object. These bool values could add understanding to positioning in the lifecycle of a collection of objects.

Visual example need to update graph

4.1.8 Direct Object Descendants / Ascendants number

Property	Value
User Input	None
Structure Input	Object Lineage Graph
Output	uint64 number for each direction
Function Name	TBA

Context - The goal of this feature is to understand from where the object originates and where the object leads to.

Visual example

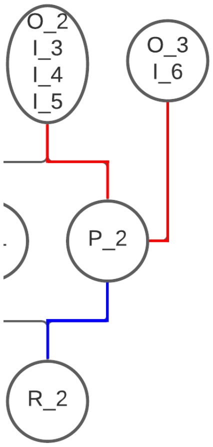


Figure 7: P_2 has two direct ascendants and one direct descendant

4.1.9 Lineage Level with total height of lineage

Property	Value
User Input	None
Structure Input	Object Lineage Graph
Output	uint64 numbers for level and total height
Function Name	TBA

Context - The goal of this feature is to understand how far through the object chain, the target object is first witnessed.

Visual example

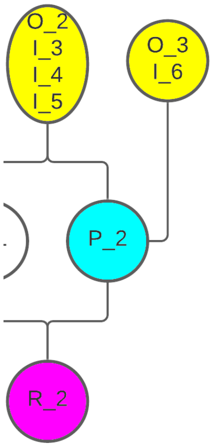


Figure 8: P_2 only has direct ancestors and neighbors. Therefore P_2 sits at level 2 out of 3

4.1.10 Object Wait time for specific event

Property	**Value**
User Input	Source Activity Name AND Target activity name
Structure Input	None
Output	uint64 time in seconds
Function Name	TBA

Context - The goal of this feature is to differentiate between items and see how long an object has to wait in order for the next step to initiate. Eg. From Figure 1: If pick_item is the source and pack_items is the target, the time i1 has to wait for pack_items to occur is far greater than i3's wait time.

Visual example

4.1.11 Object specific event directly follows

Property	**Value**
User Input	None
Structure Input	None
Output	uint64 for DF in the events (with frequency information)
Function Name	TBA

Context - The goal of this feature is to differentiate between how objects live in the ocel log. Allows to separate objects with different behaviors for further analysis

4.2 Local Features

4.3 Global Features

5 Event Based Features

5.1 Point wise Features

5.1.1 Number of objects involved separated by object type

Property	**Value**
User Input	None
Structure Input	None
Output	uint64 for each object type in log
Function Name	TBA

Context - The goal of this feature is to assign the different types of objects in a numerical way.

5.1.2 Number of objects created by event per object type

Property	**Value**
User Input	None
Structure Input	None
Output	uint64 number for each object type
Function Name	TBA

Context - The goal of this feature is to differentiate between objects that have already been seen in the log and objects that appear for the first time in the event.

5.1.3 Activity of Event

Property	**Value**
User Input	None
Structure Input	None
Output	OHE of activity names
Function Name	TBA

Context - The goal of this feature is to assign the activity name in a numerical way.

5.2 Local Features

5.3 Global Features

5.3.1 Time Series: rate of change of number of events occurring in the same time window

Property	Value
User Input	time window size
Structure Input	None
Output	float64 for each timewindow
Function Name	TBA

Context - The goal of this feature is to gather information about how active the OCEL log is.

Visual Example TODO