_coercion

Orientation Configuration and Dimensionality Markup Language

Annotation Guidelines Version 1.0

Cynthia Goodman Jasper Phillips Cory Massaro Zachary Yocum

March 7, 2013

Contents

1 Introduction

2 Extent Tags

2.1 Spatial Entity

The `spatial_entity` tag in OCDMLis intended to identify participants of spatial relations. In order to be considered a participant in a spatial relation, the entity must be located in real-space. For this annotation task, annotators should ignore any entities that exist within metaphorical spaces. For instance, a metaphrical-space is introduced in Example (1a), so, although *song* and *charts* could be considered spatial entities, they should be ignored for this task. Note, however, that metaphorical-spaces are distinct from fictional-spaces, such as in Example (1b). For the purposes of this task, annotators should the fictional-space that is associated with the diagesis of a play, film or other literary work, to be a real-space. For instance, in Example (1b), annotators should consider *PAMELA*, *door*, and *staircase* to be spatial entities.

(1) a. The hit song is on top of the charts.

b. Enter [**PAMELA**$_{se1}$] from the [**door**$_{se1}$] in front of the [**staircase**$_{se1}$], ...

2.1.1 Spatial Entity Extents

For this task, the textual extents that should be tagged with the SPATIAL_ENTITY tag will be nouns. In terms of grammatical dependencies, only the heads of noun phrases should be captured with the SPATIAL_ENTITY tag. Spatial entities may occur in nominal or named forms, and both are valid extents for this tag type. In the case of multi-word proper names, the entire extent of the name should be included in the tag. In terms of a dependency phrase-structure, sister words and phrases of head nouns should not be included in SPATIAL_ENTITY tag extents. Refer to Example (2) in section 2.1.3 for illustrations of various SPATIAL_ENTITY extent tags.

2.1.2 Spatial Entity Attributes

The OCDMLSPATIAL_ENTITY tag inherits some attributes from other annotation schemes, but only a subset of these attributes are relevant for this task. The attributes which annotators should annotate for this task are `dimensionality`, `line_type`, `area_type`, `volume_type`, `left_right`, `front_back`, `top_bottom`, and `c-sec_axis`. Annotators do not need to tag the other attributes that are defined in the Document Task Definition, including `form`, `latLong`, `mod`, `countable`, `amount`, `quant`, `scopes`.

`dimensionality` This attribute is used to designate the number of spatial dimensions occupied by the entity. For the purposes of OCDMLannotation, take spatial entities to be point-sets. As such, a value of POINT indicates a 0-dimensional entity consisting of a single point with no edges or surfaces. A value of LINE indicates a 1-dimensional entity with up to two bounding points, or vertices, with a single edge and no surfaces. A value of AREA indicates a 2-dimensional entity with some number of bounding edges, points and two surfaces. A value

Attribute	Value
id	se1, se2, se3, ...
dimensionality	POINT, LINE, AREA, VOLUME
mod	A spatially relevant modifier
line_type	SEGMENT, RAY, LINE, LOOP, OTHER
area_type	3-GON, 4-GON, DISC, ANNULUS, OTHER
volume_type	TRI_PRISM, RECT_PRISM, PYRAMID, SPHERE, TORUS, CYLINDER, CONE, BIPED, QUADRUPED, OTHER
left_right	INTRINSIC or RELATIVE
front_back	INTRINSIC or RELATIVE
top_bottom	INTRINSIC or RELATIVE
c-sec_axis	LEFT_RIGHT or FRONT_BACK or TOP_BOTTOM or OTHER

Table 1: SPATIAL_ENTITY Attributes

of VOLUME indicates a 3-dimensional entity with at least one surface and possibly a number of bounding edges and points.

If a value of POINT is specified for a SPATIAL_ENTITY tag, then it is not necessary to specify the other attributes, since they are not relevant to 0-dimensional entities. If a value of LINE is specified, then the `line_type` attribute must be filled. If a value of AREA is specified then at least the `area_type` attribute must be filled, and the `line_type` could be filled with whatever would be appropriate if the entity were coerced to a line for the purposes of participating in a relation. Finally, if the the VOLUME type is specified, then at least the `volume_type` attribute must also be filled, and the `area_type` and `line_type` attributes might also be filled.

Note: Although annotators must choose a single value for the `dimensionality` attribute, spatial entities may be coerced to different dimensionalities depending on the spatial relations they are participating in. E.g., a *door* might be considered volumetric, however it may be coerced to 2-dimensions when participating in some relationships. In that case, it would be appropriate to fill both the `volume_type` and `area_type` attributes. Refer to Section 3.1 for more discussion of this type of coercion.

`line_type` This attribute is used to classify the entity based on a set of 1-dimensional primitive types. Some lexical items that are good candidates for the `line_type` value of SEGMENT would be a *clothesline*, a piece of *wire*, or a singular strand of *hair*. These are examples of linear entities which have distinguishable endpoints. We include the RAY type to capture entities such as a *ray* of light that would have only one distinguishable endpoint. The LINE type would be appropriate for a *row* or *queue* that has no particularly distinguishable endpoints, but also is not a loop. The LOOP type would be appropriate for an *equator* or for the *border* of a region, which has no endpoints, but also loops back on itself. A value of OTHER may be specified when none of the previously mentioned values are appropriate.

`area_type` This attribute is used to classify the entity based on a set of 2-dimensional primitive types. The 3-GON type is used to indicate that the entity is triangular, possessing three distinguishable bounding edges and points such as a triangular *slice* of pizza. Similarly, the 4-GON type is used to indicate a 2-dimensional primitive with four bounding edges and points,

such as a rectangular *piece* of paper. The DISC type is used for 2-dimensional entities with a single bounding edge, and no distinguishable points, such as a round *coaster*. The ANNULUS type is appropriate for a 2-dimensional entity, such as a compact *disc*, that has two distinguishable bounding edges and no distinguishable bounding points. A value of OTHER may be specified when none of the previously mentioned values are appropriate.

`volume_type` This attribute is used to classify the entity based on a set of 3-dimensional primitive types.

The TRI_PRISM type should be used for 3-dimensional entities which can be thought of as 2-dimensional 3-gons that are extruded into a 3-dimensional prism with five distinguishable bounding surfaces and six distinguisable bounding points. A *box* of Toblerone chocolates would be an appropriate entity to be tagged with the TRI_PRISM type.

The RECT_PRISM type is used for 3-dimensional entities that are 3-dimensionally extruded 4-gons with six distinguishable bounding surfaces and eight boinding points. This type includes entities such as a standard six-sided *die*, a closed *book* or *tome*, or a prototypical *box*.

A value of PYRAMID is used for 3-dimensional entities with five distinguishable bounding surfaces, including a 4-gon and four 3-gons, in addition to five bounding points, including a single apex point.

A value of SPHERE is given to 3-dimensional spheroidal entities with no distinguishable bounding edges or vertices, such as a *globe*, *orange*, or *ball*.

A value of TORUS is used for 3-dimensional entities with no distinguishable bounding edges or vertices, but is topologically distinct from a SPHERE type by virtue of having a hole. The distinction is analogous to the difference between the 2-dimensional DISC and ANNULUS primitive types, with SPHERE corresponding to the former and TORUS corresponding to the latter. A *doughnut*, a hoola *hoop*, or a wedding *band* would all be examples of entities that would take a value of TORUS for their `volume_type`.

A value of CYLINDER indicates a 3-dimensional entity with three distinguishable surfaces, two distinguishable bounding edges, and zero vertices. A soda *can*, a dumbbell *bar*, and a AA *battery* are all examples that fall under this volume type.

A value of CONE should be given for 3-dimensional entities with two distinguishable surfaces, one bounding edge, and a single apex point. An ice-cream *cone*, a *funnel*, or a coniferous *tree* would all be appropriate entities to annotate with the CONE type.

The `volume_type` attribute is not intended to capture every entity perfectly. Rather, it is intended to identify distinguishable topological features which are accessed when the entity participates within a spatial configuration relation. If none of the previously described values are appropriate, a value of OTHER may be specified.

`left_right` This attribute is a bit that indicates whether the entity possesses an intrinsic axis of orientation whose polar extremes are 'left' and 'right'. An entity should be considered to possess an intrinsic left-right axis if it has left-hand and right-hand sides that are distinguishable independent of any frame of reference. One heuristic which may help to determine whether a spatial entity possesses an intrinsic `left_right` axis is the linear-array-test. The linear-array-test can be employed by asking "Could duplicates of this entity be arranged in a 1-dimensional array from 'left-to-right', i.e., such that the 'left' boundary belonging to each entity in the array abuts the 'right' boundary of another?" If the answer is "no", then the `left_right` attribute should probably be annotated as RELATIVE; if "yes", the value would be likely be INTRINSIC.

`front_back` This attribute is similar to the `left_right` attribute. A value of INTRINSIC indicates the entity possesses an intrinsic axis of orientation whose polar extremes are 'front' and 'back'. A value of RELATIVE indicates the opposite. The linear-array-test heuristic can be modified for this case such that arrangement of the array would be considered 'front-to-back'.

`top_bottom` This attribute is similar, again, to both `left_right` and `front_back`. A value of INTRINSIC indicates the entity possesses an intrinsic axis of orientation whose polar extremes are 'top' and 'bottom'. The linear-array-test can be applied for this attribute as well to test if the entities can possibly be stacked 'top-to-bottom'.

`c-sec_axis` This attribute is intended to identify the salient cross-sectional axis for 3-dimensional entities. The point of specifying the cross-sectional axis is to distinguish between spatial entities such as a typical twelve-ounce *can* of soda, and a military-style *submarine* that are both cylindrical, and both possess intrinsic left, right, front, back, top, and bottom boundaries, yet whose salient cross-sectional axes do not correspond to one another. For the soda *can*, the salient cross-sectional axis—the axis along which the 2-dimensional circular base would be extruded—is the TOP_BOTTOM axis. For the *submarine*, contrastively, it is the FRONT_BACK axis which corresponds to the cross-sectional axis along which the cylindrical primitive form would be extruded. Another way to conceptualize the salient cross-sectional axis would be to imagine slicing the entity into pieces as if to skewer the pieces like a kebab. Under this conceptualization, the axis that is aligned with the imaginary kebab is the axis which should be filled for the `c-sec_axis` attribute.

2.1.3 Spatial Entity Examples

(2) a.

2.2 Orientation Signal Tag

An orientation signal is taken to be some word or phrase that restricts the potential configurational interpretations of a spatial entity or group of spatial entities. Orientation signals always trigger configuration links, so if an annotator tags a orientation signal, then they are committing to also creating a configuration link.

Examples of the former type are enumerated in example (3).

(3) a. On the large table *stands* a lighted lamp.

b. . . . *lies* down on sofa.

Here, in example (3a), the signal *stands* serves to adumbrate in what configuration the lamp is arranged individually. Because the lamp *stands* rather than, e.g., *lies down*, one immediately recognizes that the lamp is situated in such a way that its cross-sectional, top-bottom axis is aligned vertically.

Likewise, in example (3b), the signal *lies* indicates that the entity—in this case an implicit spatial entity, which would need to be created via a non-consuming extent tag—is configured in such a way that its salient cross-sectional axis is horizontally aligned.

Examples of the latter type are enumerated in example (4).

(4) a. MAURICE and HENRIETTE are in evening dress and sit facing each other ...

b. In the foreground a table is spread, with flowers in the centre, bowls full of fruit, wine in decanters, oysters on platters ...

2.2.1 Orientation Signal Extents

In general, whereas orientation entities will be nominals, orientation signals will belong to other parts of speech, e.g., verbs, prepositions, adjectives, and adverbs. For phrasal verbs, take only the head verb as the extent for the orientation signal.

2.2.2 Orientation Signal Attributes

The orientation signal tag has one attribute that annotators need to consider, as given in table 2:

Attribute	Value
`id`	`os1`, `os2`, `os3`, ...
`orientation_type`	LATITUDINAL, LONGITUDINAL, LATERAL, VERTICAL, OTHER

Table 2: Attributes for ORIENTATION_SIGNAL

`orientation_type` The `orientation_type` attribute specifies an axis of orientation that is accessed by the orientation signal. It is employed to indicate how an entity is configured in an absolute sense (i.e., independently of any specific ground). The LATITUDINAL value refers to the left-right axis, the LONGITUDINAL to the front-back axis, and the VERTICAL to the top-bottom axis. We supply also the LATERAL value for cases in which a horizontal axis is accessed, but there is no distinction between the left-right and the front-back. The value of OTHER should be selected for such cases as specify a configuration not covered by the remaining values.

2.2.3 Orientation Signal Examples

Provided below, in example (5), are sample annotations for extents to be tagged as ORIENTATION_SIGNAL:

(5) a. On the large table [**stands**$_{os1}$] a lighted lamp.

b. ... [**lies**$_{os2}$] down on sofa.

c. MAURICE and HENRIETTE are in evening dress and [**sit**$_{os3}$] [**facing**$_{os4}$] each other ...

d. In the foreground a table is spread, with flowers [**in**$_{os5}$] the centre, bowls [**full**$_{os6}$] of fruit, wine [**in**$_{os7}$] decanters, oysters [**on**$_{os8}$] platters ...

3 Link Tags

3.1 Configuration Link Tag

The CONFIGURATION_LINK relates two SPATIAL_ENTITY tags. For each CONFIGURATION_LINK, one SPATIAL_ENTITY will be referred to as the figure in the relation, and the other will be referred to as the ground. The figure is the spacial entity that moves or is located in relation to the ground. Similarly, the ground is the spacial entity against which the figure moves, or in relation to which the figure is located. In example (6a), 'cat' is the figure and 'table' is the ground, since the cat is located in relation to the table. In example (6b), 'man' is the figure and 'street' is the ground, since the man is moving in relation to the street. The figure is typically the smaller, more movable object, but this is not always the case, as can be seen in example (6c). In English, the figure tends to come before the ground in the sentence. While this is a helpful fact, it should not be the sole basis of your annotation decisions.

(6) a. The cat is sitting on the table.

b. The man walked down the street.

c. The house landed on the witch.

d. ...sitting at the desk was a beautiful woman.

3.1.1 Configuration Link Attributes

The attributes are enumerated in table 3.

Attribute	**Value**
`id`	`cl1`, `cl2`, `cl3`, ...
`figure_config`	LEFT, RIGHT, FRONT, BACK, TOP, BOTTOM, OTHER
`ground_config`	LEFT, RIGHT, FRONT, BACK, TOP, BOTTOM, OTHER
`dim_coercion`	FIGURE, GROUND, or NONE
`trigger`	An ID of a `orientation_signal` tag that triggers the link
`direction`	FIGURE_GROUND, GROUND_FIGURE

Table 3: Attributes for CONFIGURATION_LINK

The combination of the `figure_config` and `ground_config` attributes are used to specify the configuration between the participating figure and ground entities. Looking back at example (6a), the configuration of the entities involved is such that the bottom of the cat is on the top of the table. The `figure_config` attribute is compulsory whereas the ground config attribute is optional; In example (7a) the `ground_config` is not explicit, whereas in example (7b) the `ground_config` is requisite.

(7) a. The dog sat opposite the man.

b. The dog sat on the man's lap.

`dim_coercion` This attribute is used to indicate that the dimensionality of the figure entity is coerced to that of the ground or vice versa. Consider example (8a). While we know that the sun is a 3-dimensional sphere, in this example the sun is being coerced into 2-dimensions by the ground, namely the sky. This type of coercion does not take place in example (8b), since both the earth and the sun are 3-dimensional objects moving in 3-dimensional space.

(8) a. The sun travels across the sky.

b. The earth rotates around the sun.

`trigger` This attribute takes the ID value of the `orientation_signal` that triggered the `configuration_link`. Since creating an `orientation_signal` automatically commits you to creating a `configuration_link`, each link is required to have a value for the `trigger` attribute. Take example (5a), the ORIENTATION_SIGNAL [**stands**$_{os1}$] is the `trigger` for the CONFIGURATION_LINK between *table* and *lamp*.

`direction` This attribute is used to indicate the asymmetrical directionality of the relation in terms of figure-to-ground or ground-to-figure.

3.1.2 Configuration Link Examples

Provided below, in example (9), are examples of CONFIGURATION_LINKS.

(9)

4 Fully Annotated Examples