ReCogLab Task Field Documentation

This doc describes different fields accessible via generate_dataset_v1.ipynb notebook.

For issues encountered with running the benchmark, please contact:
{ahliu, hprior, gargisb, kmarino}@google.com

Global Configuration

- `domain`: The specific task to generate a problem on. Will call the configuration for each specific task like Social Network and Comparison.
- `dataset_name`: An additional label to be used when saving the tfrecord file
- `recoglab_dataset_dir`: The folder to save the tfrecord to
- `split`: Which split of entities to use to populate the example
- `num_examples`: The number of examples to attempt to generate. When heuristic rebalancing is disabled this will generate num_example dataset. When Heuristic rebalancing is enabled, this will generate num_example dataset and reject some of them.
- `csv_seeds`: Comma-separated values of seeds to generate datasets for

Common Options

- `num_entities`: The number of entities to use when constructing the graph problem
- `graph_type`: The graph structure used for Comparison and Social Network
- `ordering`: The premise order of the statements. Inorder and reverse follow the topological sort of the graph.
- `use_heuristic_rebalance`: If enabled, attempts to reject sample heuristically based on the **example metadata** field heuristic_rebalance_field
- `heuristic_rebalance_field`: The metadata field to apply negative rejection sampling too. If a particular metadata value occurs frequently, we will discard future generations to eliminate common occurring metadata
- `add_filler`: Whether or not to add extra filler to context
- `num_filler_lines`: How many extra lines to add
- `filler_type`: What type of filler to add
  - random_text: random english sentences
  - entity_filler: adds sentences which include context entities but doesn't affect reasoning logic.
- `filler_position`: How filler text will be merged with the problem context.

Tasks Specific Options

Social Network Options

- `task_type`: Different FastestMessage problems
- `entity_type`: The source of entities to draw from
    - baby-names
- `relation_type`: The source to render text for the relationship. friend_advanced is flavor text

Comparison Options

- `task_type`: Comparison is the standard comparison problem, ConsistencyDetection will create problems that check if the premises are consistent, FeasibilityDetection will create problems whose questions are potentially infeasible to answer.
- `entity_type`: The source of entities to draw from
    - basic_objects: data/physical_objects_entities.csv
    - baby-names
    - congruent_objects: data/physical_objects_entities.csv with height/weight annotations
    - random_name: samples a name from a random index in data/random_string.txt
- `relation_type`: The kind of relationship between different objects.
- `congruency_mode`: only used if congruent_objects is the entity_type. Attempts to fill in a graph problem according to the congruency described by data/physical_objects_entities.csv

Syllogisms

- `entity_type`: Currently only has one option, plural_nouns, but is a no-op if setting congruent or incongruent in which case entities are automatically generated.
- `entities_mode`: How to construct the entities in each relationship
    - preset: Use the preset entities from entity_type
    - congruent: Generate entities such that each relationship is congruent with real-world knowledge. e.g. “All corgis are dogs”
    - incongruent: Generate entities such that each relationship is incongruent with real-world knowledge. e.g. “All dogs are corgis”

Family JSON

- `task`: The task to generate Family JSON dataset for. Currently supports the following:
    - family_size: Finding the size of a given family in the context

    - family_member_hobby: Checking if a given hobby is a hobby of a specific member from a specific family
    - family_size_comparison:Comparing size of two given families
    - family_member_age_comparison:Comparing age of two members from two given families
    - family_member_hobby_comparison:Comparing hobbies of two members from two given families
- `num_families`: Number of families to include in the context per sample
- `max_members`: Maximum number of members to include per family